

JULI 1984 ÖS 50,–/Sfr 6,– DM 6,–

7|84

DAS MAGAZIN FÜR COMPUTER-FANS

Die Welt steht Ihnen offen – per Telefon

Hallo Computer, hier spricht der Commodore...

Alles über Modems und Kommunikationssoftware

Anwendung des Monats

Blumengießen mit 64

Für Epson-Drucker

Vergleichstest Centronics-Schnittstellen

"Fremdsprachen" (1)

Was bringen Forth und Pascal?

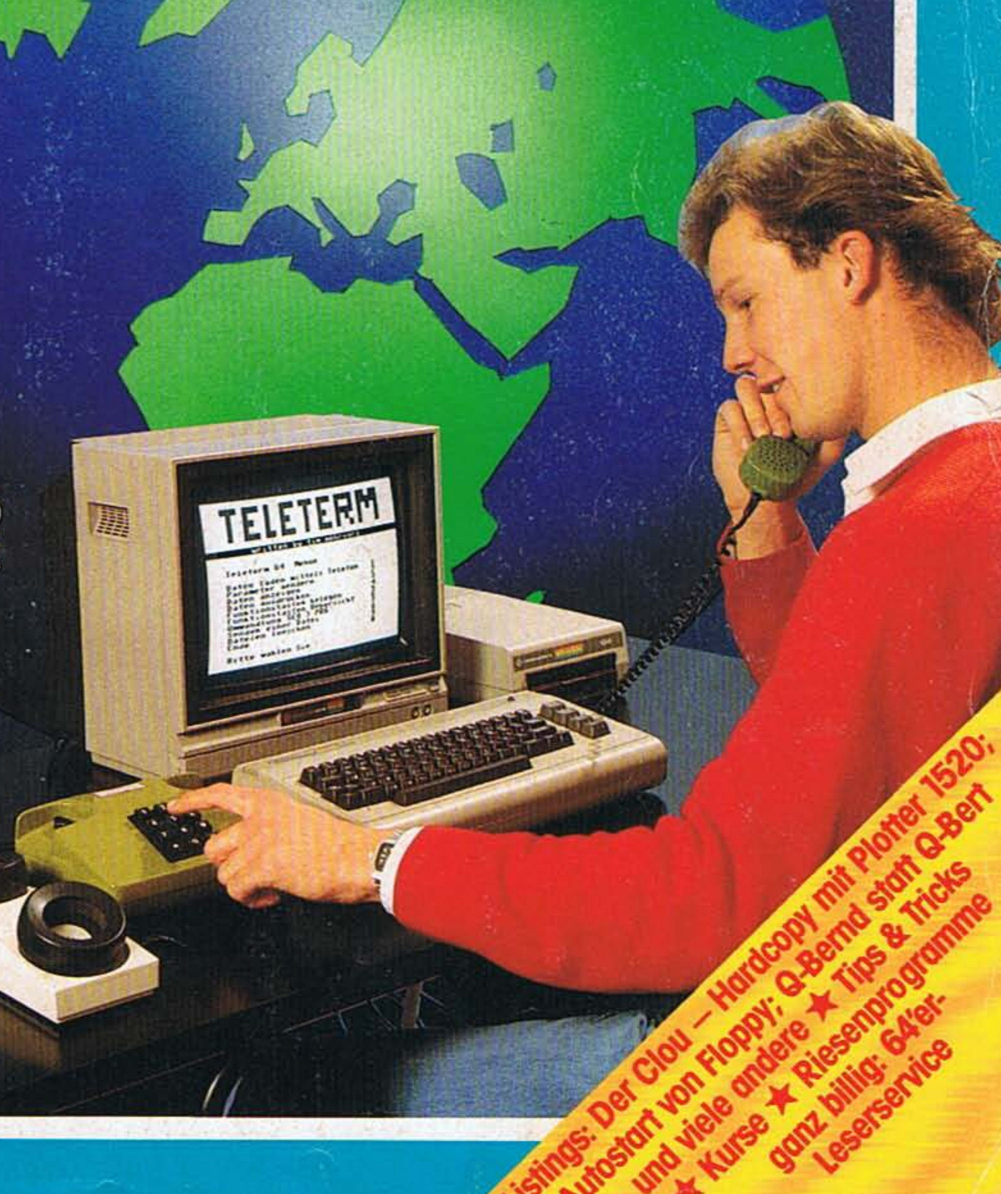
So holt man Software vom Apple auf den 64

Adressenvergleich VC 20/64

So schreibt man Programme um

Softwaretests

Magic Desk: Büro am Bildschirm?
Programmgenerator Basic Bär



Listings: Der Clou – Hardcopy mit Plotter 1520; Autostart von Floppy; Q-Bernd statt Q-Bert und viele andere ★ Tips & Tricks ★ Kurse ★ Riesenprogramme ganz billig: 64'er-Leserservice

Eine Publikation der Markt & Technik Verlag Aktiengesellschaft

INHALT



Flight II — fast wie richtiges Fliegen 68

Die Auflösung des Wettbewerbs »Das schönste Sprite« aus 64'er/4. 175

Die Anwendung des Monats. Sie kommen aus dem Urlaub und Ihre Blumen sind vertrocknet. Mit dem Commodore 64 wäre das nicht passiert. 82



Interfaces für Centronics-Drucker: Unterschiedlich in Preis und Leistung. 12

Akustikkoppler und Modems sind nicht nur für Hacker ein Schlagwort. 20

Das Listing des Monats. Sie kennen Q-Bert? Dann werden Sie von Q-Bernd begeistert sein. 139



Steuerprobleme?

Um eine Steuerung oder Regelung zu automatisieren, reicht in sehr vielen Fällen ein kleiner Computer. Schon die Hardware eines VC20-Grundmodells würde in vielen Fällen ausreichen, ein Commodore 64 wäre häufig gar nicht ausgelastet. Der gute alte 2001, in englischsprachigen Ländern »Pet« genannt, hinkte in puncto Leistung hinter den heutigen Heimcomputern deutlich her — wurde aber von vielen Technikern für Steuer- und Regelaufgaben eingesetzt. Manche dieser Systeme tun heute noch ihren Dienst.

Die Zahl der Anwendungen auf diesem Gebiet hat allerdings lange nicht so stark zugenommen, wie die Zahl der kleinen und kleinsten Computer. Das mag zu einem Teil daran liegen, daß es hier kaum vernünftige Bausätze gibt — und daß andererseits auch passionierte Bastler schnell auf Beschaffungsschwierigkeiten stoßen, wenn sie bestimmte Teile oder Baugruppen für die Realisierung der einen oder anderen Idee suchen: Wer sich mit Programmierung oder Elektronik auskennt, weiß nicht ohne weiteres, wo man welche Pumpe kaufen kann, um etwa das Blumengießen zu automatisieren.

Vorläufig war wohl noch ein ganz anderer Punkt hinderlich: Wer einen Computer hat, will ihn ja in der Regel für verschiedene Zwecke einsetzen und nicht nur zur Erledigung einer Aufgabe. Wenn der Computer aber beispielsweise die Heizung steuern soll, dann kann er für nichts anderes verwendet werden. Je mehr Benutzer im Laufe der Zeit jedoch »aufsteigen«, desto häufiger werden preisgünstige gebrauchte VC 20 zu haben sein, die der Commodore 64-Besitzer dann als Zweitgerät für einen speziellen Zweck reservieren kann. Damit wäre kein Problem den »Kleinen« auf Dauer als Zentrale einer Alarmanlage oder ...oder...abstellen. Vielleicht bekommen damit auch die Steuer- und Regel-Anwendungen etwas Auftrieb.

Michael Pauly, **Chefredakteur**



Datenübertragung ... Tagesordnung, in Deut... Ein Kaufhaus in München ... zwischen einem Apple und ein...

Modem-Show

Wir waren schon recht früh am Ort des Geschehens. Die Geschäfte hatten noch nicht lange geöffnet, dennoch herrschte bereits reger Betrieb. Unser Weg führte zu einem Kaufhaus am Stachus, einem bekannten Platz in der Münchener Innenstadt, und dort direkt in die Computerabteilung.

Vorbei an Farbe sprühenden Monitoren, die mit kleinen und großen Computern verbunden waren, auf denen begeisterte Jugendliche ganz weggetreten herumhackten, hielten wir Ausschau nach einem Commodore C 64. Aber nicht nach einem normalen, sondern nach einem, der an ein Modem oder auch an einen Akustikkoppler angeschlossen war. Unsere Erwartungen wurden um einiges gedämpft, als wir nichts dergleichen zu sehen bekamen. Sollte man uns falsch informiert haben? Waren wir im falschen Kaufhaus?

Schließlich entdeckten wir ein Gerät, von dem wir annahmen, zu Recht annahmen, daß es ein Modem war. Aber kein C 64 in der Nähe. Lediglich ein Apple, dessen Tastatur mit einer Plastikhaube verdeckt war, stand an der nächsten Regalecke (Bild 1). Dann bemerkten wir auch die Kabelverbindung zwischen diesen beiden Geräten.

Auch ein Bediener war nicht zu sehen. Selbst die Computer-Fans wußten mit dieser Ecke nicht viel anzufangen. Daß sich hier in Kürze erstaunliches abspielen sollte, ahnte sicherlich keiner von ihnen.

Wir hatten kaum Zeit, uns das Modem, dieses für Eingeweihte so interessante Gerät, genauer anzuschauen, als ein Mann auftauchte, der sich sofort daran zu schaffen machte. Nachdem wir uns vergewissert hatten, an den Richtigen gekommen zu sein, stellten wir uns vor und dann die alles einleitende Frage: »So, daß also ist das Modem ?!«



Bild 1. Der Apple war an das Modem angeschlossen

Im folgenden Gespräch erfuhren wir mehr über das, was wir zu sehen bekommen sollten. Es war erstens geplant, eine Verbindung zwischen einem C 64 und dem schon erwähnten Apple herzustellen. Zum anderen sollte eine Verbindung mit einem Computer in Köln geschaffen werden.

Apple-Programme mit dem C 64 editieren

Der Commodore 64, mit dem der Datenaustausch laufen sollte, stand an einem anderen Stand, etliche Meter weiter. Angeschlossen waren ein Diskettenlaufwerk VC 1541, ein Monitor und ein Akustikkoppler (Bild 2).

So, jetzt konnte es losgehen. Nachdem noch einmal alle Verbindungen geprüft waren, wurde zuerst die notwendige Software geladen. Sie ermöglicht es, alle Daten über die RS232-Schnittstelle über den Userport an den Akustikkoppler zu schicken, verarbeitet einkommende Daten und speichert den gesamten Dialog auf Wunsch auf Diskette.

Jetzt fehlte nur noch eines: Die Telefonverbindung. Also wurde die Nebenstelle angewählt, unter der das Modem angeschlossen war. Kurze Zeit später meldete sich der Apple! Natürlich vergewisserten wir uns sofort, ob das gleiche Bild auf dem Apple-Monitor zu sehen war. Und tatsächlich, alles lief synchron. Als nächstes wurde demonstriert, wie der C 64 Programme vom Apple holte. Wir ließen

er Telefon ist in Amerika schon fast an der chland eine Sache von wenigen Spezialisten. emonstrierte jetzt einen Datenaustausch m C 64 über Modem und Akustikkoppler.

im Kaufhaus



Bild 2. Der C 64 war über den Akustikkoppler an das Telefonnetz angeschlossen

uns Basic-Programme aus dem Apple-Laufwerk laden und konnten sie am C 64-Monitor listen und auch ändern. Auch das Zurückspeichern verlief ohne Komplikationen.

Ein ungewohntes Bild war es schon. Apple Basic auf dem C 64, ohne das die VC 1541 aktiv war. Natürlich lief alles sehr langsam ab, nämlich mit 300 Baud (Bit pro Sekunde). Das ist die Geschwindigkeit, mit der die Daten per Akustikkoppler über die RS232-Schnittstelle fließen. Aber es war sehr beeindruckend.

Da alles über das normale Telefonnetz lief, stand auch einer weiter entfernten Verbindung nichts mehr im Weg. Aber sehen wollten wir das schon. Und so konnte eine weitere Aktion ablaufen: In Köln stand ein C 64, der darauf wartete, mit dem in München stehenden Apple zu kommunizieren.

Wir brauchten nur noch auf den Anruf des Kölner Gegenüber zu warten. Unser Apple stand auf Empfang. Gespannt warteten wir darauf, die ersten Zeichen auf dem Monitor zu sehen. Und auf einmal kamen sie! Wir erlebten die Versuche des Kölner mit, mit dem Apple Kontakt aufzunehmen. Jede Taste, die er in Köln drückte, war bei uns zu sehen. Er versuchte eine Meldung für ihn abzurufen. Leider hatten wir in seinen »Briefkasten« keine Nachricht hineingeschrieben. Man merkte ihm seine Enttäuschung an. Aber wir übermittelten ihm dann direkt unsere Grüße zum Gelingen dieses interessanten Versuches.

Mit normalen Mailboxen (Briefkästen) kann man auf drei Arten kommunizieren. Erstens ist es möglich Nachrichten einzugeben. Diese Nachrichten kann jeder, der einen Akustikkoppler besitzt, eingeben und, das ist die zweite Funktion, auch abrufen und lesen. Die dritte Möglichkeit ist die direkte Verbindung zwischen Anrufer und Empfänger. Der Empfänger ist immer der Computer, der die Mailbox unterhält. In unserem Fall war das der Apple im Münchener Kaufhaus. Manche Mailbox-Programme lassen es auch zu, persönliche Nachrichten einzugeben, die nur von einem bestimmten Teilnehmer gelesen werden können. Dieser Teilnehmer muß dazu einen persönlichen »Briefkasten« in dem Mailbox-System besitzen, für den er in der Regel einen geringen Beitrag bezahlt. Er erhält eine Geheimzahl, ein Paßwort, das nur er kennt. Somit wird sichergestellt, daß nur er die für ihn bestimmten Nachrichten lesen kann.

Wir »sprachen« dann noch etwas mit Köln und beendeten die Verbindung. Die Demonstration war gelungen und wir um einiges Wissen reicher. Natürlich juckte es uns in den Fingern, diese Erkenntnisse auch selbst zu verwenden. Über unsere Erfahrungen werden wir noch ausführlich berichten.

Als wir das Kaufhaus verließen, standen die Spiele-Freaks immer noch vor ihren Computern und hatten keine Ahnung, was ihnen entgangen war... (gk)

Tragbarer Meß- und Steuercomputer SX 64 ADS

Der Meß- und Steuercomputer SX 64 ADS von Datalog bietet als kompakte Einheit die preisgünstige Möglichkeit, anfallende Meßwerterfassungs-, Regel- und Steueraufgaben an wechselnden Einsatzorten ohne großen Geräteaufwand zu lösen. Das integrierte Meßdateninterface besitzt vier Analog-Eingänge mit einer Auflösung von 12 Bit, zwei Analog-Ausgänge für Regelzwecke, vier Digital-TTL- und vier Relais-Ausgänge für Steuerungsanwendungen. Die Datenrate beträgt maximal 50 Messungen pro Sekunde zum Abfragen und Setzen aller Ein- und Ausgänge. Die Programmierung des eingebauten Interfaces geschieht direkt in Commodore-Basic und sei auch von einem wenig erfahrenen Anwender mit Basic-Kenntnissen innerhalb von Minuten zu erlernen.



Weitere Vorzüge des Kompaktsystems sind der 5-Zoll-Farbbildschirm mit der Möglichkeit, hochauflösende Grafiken (Meßwertkurven) darzustellen, sowie die eingebaute Floppy-Station mit 170 KByte Speicherkapazität. Der Computer selbst ist mit einem RAM-Speicher von 64 KByte ausreichend bestückt. Optionell liefert Datalog ein passendes RS232C (V.24)-Interface zur Kommunikation mit anderen Systemen. Insbesondere wird darauf hingewiesen, daß das komplette Software-Angebot für den weit verbreiteten C 64-Mikrocomputer ohne Einschränkungen auf dem Meß- und Steuercomputer SX 64 ADS lauffähig ist. Somit ist das System nicht nur auf dem Einsatz im technisch-wissenschaftlichen Bereich begrenzt. Umfangreiche Demo-Software sei im Lieferumfang enthalten.



Computer Journal gesucht

Wer hat noch Hefte der Jahrgänge 82/83 vom inzwischen eingestellten Computer Journal und kann mir diese ausleihen? Wer besitzt eine Anleitung zum Music Composer für den VC 20?

David Twigg-Flesner

CBM-Peripherie am Commodore 64?

Ich besitze einen CBM 3032 mit Diskettenlaufwerk und Drucker. Beim Kauf eines Commodore 64 möchte ich diese Peripherie gerne mittels eines IEC-Bus-Interfaces weiterverwenden. Hat schon jemand Erfahrungen mit solchen Interfaces gesammelt, arbeiten sie zuverlässig, und wie ist es um die mechanische Stabilität bestellt?

Oliver Fischer

Disketten beidseitig verwenden?

Wenn man mit einem Bürolocher auf einer einseitig beschreibbaren Diskette am linken Rand eine Stanzung in Höhe des Schreibschutzes am rechten Rand vornimmt, läßt sich die Diskette beidseitig auf einer Floppy-Disk 1541 verwenden. Ist die Datensicherheit dabei gewährleistet, oder spricht etwas gegen diese Methode?

Dipl.-Ing. M. Lohse

Die Floppy VC 1541 interessiert sich nicht für das kleine Indexloch in der Diskette. Andere Floppystationen benutzen dieses Loch zur internen Steuerung und Synchronisation. Deshalb funktioniert der Trick bei solchen Laufwerken nicht. Bei der VC 1541 gibt es jedoch keine Probleme. Allerdings ist bei einseitig beschreibbaren Disketten auch nur eine Seite durch den Hersteller geprüft. Die zweite Seite kann daher unter Umständen fehlerhafte Sektoren enthalten, die aber vom Floppy-Betriebssystem dann nicht benutzt werden. Wenn Sie nach dem Formatieren beim Listen der Directory die Meldung »664 Blocks free« sehen, sollte jedoch die Datensicherheit einigermaßen gewährleistet sein.

Für sehr wichtige Aufzeichnungen empfiehlt sich jedoch immer die Verwendung doppelseitig geprüfter Disketten.

Probleme mit Speichererweiterung

Neulich habe ich mir eine 16-KByte-RAM-Erweiterung für meinen VC 20 gekauft. Als ich aber versuchte, ein Spiel, das für 3 KByte-RAM und einen Joystick gedacht ist, laufen zu lassen, funktionierte es nicht, obwohl ich einen Joystick habe. Wer kann mir helfen?

Daniel Hüller

Beim VC 20 liegt der Bildschirmspeicher je nach RAM-Erweiterung in verschiedenen Adreßbereichen. Daher laufen viele Programme für die Grundversion nicht mit Speichererweiterung. Einfache Lösung für das Problem: Das Steckmodul entfernen.

Spielregeln

Wir verschicken keine Prospekte oder ähnliche Produktinformationen — die müssen Sie direkt beim Lieferanten des Produktes anfordern; die Anschrift kann bei uns erfragt werden.

Wir können keine Programme umschreiben oder anpassen. Wenn ein Leser ein von uns veröffentlichtes Programm umgeschrieben hat und bereit ist, das Listing abzugeben, können wir einen entsprechenden Hinweis im Leserforum veröffentlichen.

Ob und wann Antworten auf die veröffentlichten Fragen eingehen, läßt sich nicht voraussagen; wir sind nicht in der Lage, Vormerklisten zu führen und einzelne Leser individuell zu informieren, wenn eine Antwort eingegangen ist. Wir sind aber gern bereit, den Kontakt zwischen Lesern herzustellen, die am gleichen Thema interessiert sind.

Fragen Sie doch!

Selbst bei sorgfältiger Lektüre von Handbüchern und Programmbeschreibungen bleiben beim Anwender immer wieder Fragen offen. Viel mehr Fragen ergeben sich bei Computer-Interessenten, die noch keine festen Kontakte zu Händlern, Herstellern oder Computerclubs haben. Sie können der Redaktion Ihre Fragen schreiben oder Probleme schildern (am einfachsten auf der beigehefteten Karte). Wir veranlassen, daß die Fragen von einem Fachmann beantwortet werden. Allgemein interessierende Fragen und Antworten werden veröffentlicht.

RESET über User-Port?

Um bei meinem Commodore 64 nach einem Spiel ein anderes Programm zu laden, muß in den meisten Fällen der Computer aus- und dann wieder eingeschaltet werden. Kann man das umgehen, indem man kurzzeitig Pin 1 und 3 des User-Ports miteinder verbindet, oder führt dies zu einer Beschädigung des Computers?

Walter Stiller

Pin 1 des User-Ports liegt an Masse, Pin 3 ist die RESET-Leitung. Stellt man kurzfristig eine Überbrückung her (zum Beispiel mit einer Büroklammer oder ähnlichem), dann wird ein RESET ausgelöst, wie der Computer ihn auch nach dem Einschalten durchführt. Der RESET an sich ist ungefährlich. Falls man jedoch beim Manipulieren am User-Port unabsichtlich andere Pins miteinander in Verbindung bringt, kann dies zu Beschädigungen des VIA-Bausteins führen. Es empfiehlt sich daher den Einsatz eines kleinen Tastschalters.

In der Ausgabe 8/84 zeigen wir übrigens verschiedene Möglichkeiten, Reset-Tasten anzubringen.

Nochmals DOS 5.1

Der Artikel über das DOS 5.1 in der Ausgabe 5/84 ist ja sehr interessant. Leider kann man bei den meisten kommerziellen Programmen aber nichts damit anfangen, da man aus diesen Programmen nur durch Abschalten des Computers wieder herauskommt. Aber dann ist auch das DOS 5.1 verloren, und es jedesmal neu einzuladen ist doch zu umständlich. Kann man da nichts dran machen?

Heinrich Carstensen

Das DOS 5.1 ist eigentlich nur bei der Programmentwicklung nützlich. Es erleichtert das Arbeiten mit der Floppy, indem es Abkürzungen verwendet. Wenn man Spiele oder andere kommerzielle Software benutzt, ist die Anwendung des DOS sowieso wenig sinnvoll.

64 KByte-Erweiterung für VC 20?

Als VC 20-Anwender möchte ich mir eventuell eine 64-KByte-RAM-Karte kaufen. Welche der angebotenen Karten ist die beste? Gibt es bei der Anwendung Probleme, zum Beispiel beim Laden von Programmen für die Grundversion? Gibt es überhaupt eine Möglichkeit, die 64 KByte voll zu nutzen, vielleicht als RAM-Disk?

Jens Bümmerstede

Wir werden in einer der nächsten Ausgaben eine 64-KByte-RAM-Karte testen. Soviel vorweg, eine Pseudo-Floppy (oder RAM-Disk) läßt sich durchaus realisieren.

Schachprogramme

Frage: Wer kennt Spielstärke Schachprogramme?
Ausgabe: 5/84

Peter Jugl

In einem aktuellen neutralen Vergleich in England wurden die folgenden Schachprogramme für den C 64 unter Turnierbedingungen (= 3 min/Zug) getestet: SARGON II, CHES 7.0, COLOSSUS 2.0 und GRANDMASTER. Jedes Programm spielte dabei gegen jedes je einmal mit weiß und schwarz. Dabei wurde folgendes Endergebnis erzielt:

Programm	Gewonnen	Remis	Verloren	Punkte
1. Grandmaster	4	2	—	5
2. Chess 7.0	2	3	1	3,5
3. Colossus 2.0	2	2	2	3
4. Sargon II	—	1	5	0,5

Wie Herr Wacker aber richtig schrieb, sind alle heutigen Schachprogramme für Homecomputer und selbst die neuesten Schachcomputer noch nicht in der Lage, gegen gute Vereinsspieler zu bestehen. Entscheidende Besserung dürfte erst mit modernerer Hardware (z.B. 16/32-Bit Prozessor) zu erwarten sein. Fritz Schäfter, Kingsoft

Komma als Satzzeichen

Frage: Wie kann ich bei dem INPUT-Befehl das Komma als Satzzeichen verwenden?
Ausgabe: 5/84

Gerhard Giessmann

Kommata bei INPUT einzugeben funktioniert durchaus, vorausgesetzt die Eingabe steht in Anführungszeichen. HALLO, WIE GEHT'S? geht nicht, »HALLO, WIE GEHT'S?« aber schon. Die Anführungszeichen stehen nachher nicht in der Variablen, sie lassen den Computer lediglich das Komma als das ansehen, was es sein soll: Teil des eingegebenen Strings. Das gleiche gilt für den Strichpunkt und den Doppelpunkt. Enthält die Eingabe selbst auch Anführungszeichen heißt es aufpassen: Das erste Anführungszeichen in der Eingabe hebt das Anfangsanführungszeichen wieder auf, ein nachfolgendes Komma würde wieder als Trennzeichen interpretiert. »ICH HEISSE »WRXL«, UND DU?« ginge durchaus. Das erste Anführungszeichen ist zwar durch das zweite aufgehoben, dann folgt jedoch noch ein drittes, das dem Computer wieder sagt: es folgt ein reiner String. »NAME: »MEIER, ANDREAS«« ginge nicht. Der Doppelpunkt wird zwar nicht stören, wohl aber das Komma. Textteile, die selbst in Anführungszeichen stehen dürfen also keine Trennzeichen enthalten (,;:).

Metin A. Savignano

Zehnertastatur für C 64

Frage: Gibt eine zusätzliche Zehnertastatur (Zifferblock) zum Anschluß an den C 64?
Ausgabe: 4/84

Arndt Grass

Die Firma Computertechnik Hartmann, Bismarckstraße 5 in 6360 Friedberg 1, Tel. 06031/14863, bietet für 89 Mark plus Versandkosten eine solche Tastatur an. Sie besitzt 20 Tasten, nämlich Zehnerblock, Punkt, Komma, Leer- und Return-Taste sowie die Buchstaben A bis F (zur Verwendung bei hexadezimaler Eingabe). Die Tastatur ist sowohl für den C 64 als auch für den VC 20 zu verwenden.

Gero Morres

Steckmodule abspeichern?

Beim VC 20 liegt der Modulbereich von $A000 bis $BFFF und läßt sich mit folgender Eingabe auf Diskette kopieren: POKE 43,0 : POKE 44,160 : POKE 45,0 : POKE 46,192 : SAVE »(Name)«,8.

Wer kann mir mitteilen, wo der Modulbereich meines Commodore 64 liegt und ob es eine ähnliche Routine zum Abspeichern dieses Bereichs auf Diskette gibt?

Hartmut Götze

Der Steckmodulbereich beim Commodore 64 liegt von $8000 bis $9FFF. In der obengenannten SAVE-Routinen müssen daher der POKE-Wert 160 durch 128 und der Wert 192 durch 160 ersetzt werden. Allerdings haben die Hersteller von Steckmodulen in der Regel einige Sicherungen gegen unerlaubtes Kopieren eingebaut.

Sprachausgabe mit VC 20

Frage: Wer kennt ein Programm zur Erzeugung von Sprache auf dem VC 20?
Ausgabe: 5/84

Georg Brandt

Die Firma Adman Electronics Ltd., Ripon Way, Harrogate N.Yorks. HG 12AU in Großbritannien bietet für den VC 20 / V 64 einen Speech-Synthesizer für englische Sprachausgabe zum Preis von 49,95 Pfund an.

Spiele dafür sind von Bugbyte (Twin Kingdom Valley), Voyager (Attack, Attack) und Thor Computer Software (3D Silicon Fish) schon im britischen Fachhandel erhältlich.

David Twigg-Flesner

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines gutes Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen beziehungsweise Produkten. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere, bessere Antwort als die hier gelesene — dann schreiben Sie uns doch. Die Antworten werden wir in einer der nächsten Ausgaben publizieren. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.

Grafik mit VC 1515

Frage: Wie kann man verhindern, daß der VC 1515 Drucker von Zeit zu Zeit mit der Fehlermeldung »DEVICE NOT PRESENT ERROR« aussteigt?
Ausgabe: 5/84

Joachim Bolle

Das Problem läßt sich auch umgehen, wenn man alle PRINT #-Anweisungen durch normale PRINT-Befehle ersetzt. Zu diesem Zweck muß die Ausgabe mit der CMD-Anweisung auf den Druckerkanal umgelegt werden.

In meinen Programmen erfolgen Ausgaben auf dem Drucker also nicht mehr so:

OPEN 4,4 : PRINT#4, "Test": CLOSE 4,

sondern in der folgenden Form:

OPEN4,4:CMD4:PRINT"Test" : PRINT#4 : CLOSE 4

Durch den Wegfall der PRINT #-Anweisungen laufen meine Programme nun völlig störungsfrei. Hans-Jürgen Stadelmann

Rechengenauigkeit

Frage: Wie erklärt sich der Unterschied in der Rechengenauigkeit von Taschenrechnern und Computern?
Ausgabe: 5/84

Albert Bartels

Ihre Antwort zur Leserfrage ist natürlich richtig. Die Rechengenauigkeit des Computers ist geringer. Allerdings wurde die Frage nur teilweise beantwortet. Das Ergebnis der Berechnung des Computers wird in Bogenmaß ausgegeben. Die meisten Taschenrechnern berechnen jedoch in Gradmaß.

Um das Ergebnis des Computers in Gradmaß zu erhalten muß folgende Umrechnung erfolgen, wobei für x der jeweilige Winkel, für pi 3.14159265 eingesetzt werden muß:

Gradmaß Bogenmaß
Sin x = Sin (x*pi/180)
Beispiel: Sin 45 = 45*3.141592
65/180)

Siegfried Dietrich

Vergleichstest

Centronics — Schnitts

Ein Interface muß die richtige Verbindung herstellen, um einen Drucker mit Centronics-Eingang am Commodore 64 zu betreiben. Mittlerweile werden einige solcher, auch Schnittstellen genannte, Erweiterungen auf dem Markt angeboten.

Doch nicht alle leisten und kosten das gleiche. Welches Interface ist das beste?



Bild 2. Das Testfeld: Hard- und Softwarelösungen

Die Tatsache, daß sich bisher kaum ein Hersteller von Heimcomputern zu einer Normung der Verbindungsports durchgerungen hat, ist sicher jedem C 64-Besitzer bekannt. Wer einmal versucht hat, einen anderen, als einen Commodore-Drucker an seinen Computer anzuschließen, wird festgestellt haben, daß dies nicht ganz unproblematisch ist. Weder am Drucker, noch am C 64 befindet sich ein Ausgang, der dem anderen auch nur im entferntesten ähnelt. Da aber gerade die Drucker von Fremdherstellern oft mit überlegenen Leistungen aufwarten können, ist ein ständig wachsender Markt von Schnittstellen verschiedenster Konstruktionsweisen entstanden.

Das Hauptunterscheidungsmerkmal bei diesen Schnittstellen ist die Art der Datenübertragung und -anpassung. Zum einen werden sogenannte Softwarelösungen angeboten, bei denen die Anpassungen der Daten mit Hilfe von ladbarer beziehungsweise auf Eproms steckbarer Software im Computer selbst vorgenommen und über den User-Port zum Drucker gesendet wird. Zum anderen gibt es die Hardwarelösungen (Bild 1), die, mit eigenem Prozessor versehen, die Datenanpassung auf einer externen Platine durchführen und ausnahmslos den seriellen Bus zur Datenübertragung verwenden.

Der Testablauf

Getestet wurden natürlich sowohl Hard- als auch Softwareschnittstellen (Bild 2). Die Vertreter der Hardwareseite waren das Görlitz-Interface, zwei ungleiche Versionen von Wiesemann und das weit verbreitete Data Becker-Interface. Auf der Softwareseite traten die Eprom-Versionen von Kalawsky, Bockstaller und die Diskettenversion der in diesem Heft beschriebenen Schnittstelle unseres Lesers Helmut Eyssele, zum Test an.

Erstes Testkriterium waren alle in den zugehörigen Bedienungsanleitungen angegebenen Funktionen. Zusätzlich mußten die Kandidaten noch zwei Sonderprüfungen ablegen: Ihre Verträglichkeit mit einer Reihe von bekannten Textverarbeitungsprogrammen (siehe Bild 3) und ihre Fähigkeiten bei der Erstellung einer Hardcopy vom Bildschirminhalt (was noch für Überraschungen sorgte). Da die einzelnen Testteilnehmer über die verschiedensten Befehle zur Ansteuerung ihrer Funktionen verfügen, war es nicht sinnvoll, ein einheitliches Testprogramm zu schreiben. Es wurde aber trotzdem versucht, ähnliche Funktionen zu vergleichen.

Der Alleskönner

Mit zirka 340 Mark nicht gerade das billigste, stellte sich das Görlitz-Interface zum Test. Diese Hardwarelösung ist zum Einbau in einen Epson MX/RX/FX 80-Drucker vorgesehen. Der Einbau ist, auch für technische Laien, problemlos, denn die Platine (Bild 1) wird lediglich in den geöffneten Drucker eingesteckt und ist sofort betriebsbereit. Der Epson wird fortan wie ein Commo-

tellen



Bild 1. Der Sieger — das Görlitz-Interface

dore-Drucker angesprochen. Die Commodore-eigenen Steuerzeichen werden dabei, wie gewohnt, als reverse Grafiksymbole ausgedruckt. Dies deutet darauf hin, daß auch die Grafikzeichen dem Drucker keine Schwierigkeiten bereiten. Der eingebaute Selbsttest (nicht der des Druckers) zeigt den gesamten neuen Zeichensatz. Wem das aber immer noch zu wenig ist, hat die Gelegenheit den gesamten CBM-Zeichensatz in bis zu vierzig Variationen aus doppelter Breite, doppelter Höhe und reverser Darstellung auszudrucken. Ferner verfügt das Görlitz-Interface über einen Grafik-Modus, in dem alle Zeichen, deren Code größer als 127 ist, als senkrechte Punktreihe aufgefaßt werden. Durch Aneinanderreihen vieler dieser Reihen können beliebige Zeichen gedruckt werden. Daß die Steuer- und Formatierungsbefehle, wie sie im Epson-Handbuch beschrieben sind, in jedem Modus erhalten bleiben, gefiel beim Test besonders.

Die letzte Station unserer Testreihe, der Bildschirmausdruck, wurde mit besonderer Spannung erwartet. Die Aufgabe lautete: Mit den in der Data Becker-Supergrafik und Simons Basic enthaltenen Hardcopyroutinen in möglichst kurzer Zeit einen genauen Abdruck des Bildschirms auf das Papier zu bringen. Nach der Vorbereitung des Interfaces durch die folgenden Basic-Zeilen:

```
100 OPEN 1,4 :REM Druckerkanal öffnen
110 PRINT#1,CHR$(27)"V":REM VCEI Grafic
120 PRINT#1,CHR$(08):REM 7-Nadel Einzelp.
130 OPEN9,4,9,"7":REM Zeilenabst. auf 0
140 Copy:REM Simons Basic Hardcopy
150 CLOSE1:CLOSE9:REM Drucker-kan.schl.
160 END
```

konnte der Ausdruck beginnen. Ein kleines Testprogramm erstellte die auszudruckende hochauflösende Grafik und der Drucker begann seine Arbeit. Nach zirka 40 Sekunden war der Bildschirminhalt auf Papier verewigt, allerdings war der Ausdruck etwas klein. Aber auch hier bietet das Görlitz VCEI eine Veränderungsmöglichkeit an: Durch Einfügen einer zusätzlichen Zeile ist eine variable Breite des Ausdrucks möglich. Bei dieser Fülle von Anwendungsmöglichkeiten erscheint es fast schon selbstverständlich, daß auch bei der Konzeption der Platine an den Anwender gedacht wurde. So ist beispielsweise der serielle Bus am Interface doppelt vorhanden, damit der Drucker nicht das letzte angeschlossene Gerät sein muß. Die mögliche Abschaltbarkeit der Schnittstelle ist dann sinnvoll, wenn der Drucker auch noch an anderen Computern eingesetzt werden soll. Man erspart sich so den Ausbau des Interfaces. Eine Besonderheit, mit der sonst keine andere der im Test befindlichen Schnittstellen aufwarten konnte, ist der eingebaute Pufferspeicher von zwei KByte. Dies ist besonders angenehm beim Arbeiten mit Textverarbeitungsprogrammen, da bereits weitergeschrieben werden kann, während der Drucker noch arbeitet. Beim Test der Kooperation mit den Textverarbeitungsprogrammen gab es in keinem Fall Probleme beim Ausdrucken der Texte.

Insgesamt machte das Görlitz-Interface einen hervorragenden Eindruck. Beim ständigen Arbeiten mit dem Drucker vergißt man nach einiger Zeit vollkommen, daß der Epson FX 80 eigentlich nicht speziell für den Commodore 64 konzipiert wurde.

Die Düsseldorfer Lösung

Für den Einbau des VCI von Data Becker gilt das gleiche, wie für das Görlitz-Interface. Leider hat man bei Data Becker vergessen, den seriellen Bus durchzuführen, so daß der Drucker das letzte Gerät sein muß. Wie aus Bild 3 ersichtlich ist, sind Listing-Modus, Grafik- und reverse Zeichen sowie der Direkt-Modus verfügbar und über Sekundäradressen einzustellen. Auch ein spezieller Grafikmodus. Die Frage, ob diese Grafikfähigkeit aber auch für einen Bildschirmausdruck genügend war, ließ die Redaktion voller Erwartungen an den Hardcopytest herangehen. Naheliegend war es natürlich, das Hilfsprogramm aus dem gleichen Hause, die Supergrafik, zu verwenden. Getreulich den Worten der Anleitung folgend, versuchten wir mit dem Befehl für Acht-Nadel-Drucker, ein Bild zu kopieren. Das Ergebnis war ernüchternd, denn der Drucker regte sich zwar, aber das was er druckte, sah aus wie Nebel über London. Erst als wir es mit dem Sieben-Nadel-Hardcopybefehl versuchten, hatten wir Erfolg. Von diesem Erlebnis angespornt versuchten wir es auch noch mit Simons Basic. Nach dem Eingeben des folgenden kleinen Programms klappte es auf Anhieb.

```
100 OPEN 1,4:REM Druckerkanal öffnen
110 PRINT#1,CHR$(8):REM Grafik Modus
```

120 PRINT #1,CHR$(26):REM Grafik Byte wiederh.
130 Copy: REM Simons Basic Hardcopy
140 CLOSE1:REM Druckerkanal schließen

Was wir nicht erwartet hatten: Es gab bei den Zeitmessungen für die Hardcopyausdrucke enorme Unterschiede zwischen den einzelnen Schnittstellen. Für das gleiche Bild benötigte das Data Becker-Interface sechsmal länger (zirka 4 Minuten) als das Görlitz-Interface. Dies ist wahrscheinlich damit zu erklären, daß durch das VCI ein Commodore 1526 simuliert wird (ohne dabei auf die Vorzüge des Epson-Druckers zu verzichten), der aber selbst mit neuem ROM nur ein frei definierbares Zeichen besitzt. Das Zusammenspiel mit den ausgewählten Textverarbeitungsprogrammen funktionierte auch mit dem VCI einwandfrei. Wer aber lieber eigene Texte eingeben, beziehungsweise in seine Programme einbauen will, wird die Funktion »Festlegen der Druckposition« des VCI bald nicht mehr missen wollen. Durch diese Funktion ist es möglich, die Punktposition, ab der ein Text auf dem Papier gedruckt werden soll, festzulegen.

Bis auf die etwas langsame Hardcopy ist dem Data Becker-Interface wenig Negatives nachzusagen, wenn es auch nicht ganz die Möglichkeiten der Görlitz-Schnittstelle bietet. Dafür ist es aber mit 298 Mark auch um 40 Mark billiger als das Görlitz-Pendant. Das ist natürlich immer noch ein stolzer Preis.

Zwei ungleiche Brüder

Die nächsten beiden Testgeräte von Wiesemann sehen zwar äußerlich vollkommen gleich aus, in ihren Leistungsmerkmalen unterscheiden sie sich aber erheblich. Das intelligentere der beiden hat den Namen VC 20/CBM 64-Interface Typ 9200 NEC und ist für NEC und kompatible Drucker vorgesehen. Sein Bruder verzichtet auf den Zusatz NEC im Namen und auch auf so manche Fähigkeit. Er ist für die Ansteuerung eines Epson-Druckers vorgesehen. Die Installation dieser Schnittstellen ist vorbildlich einfach, denn sie werden nur in den Eingang des Druckers, beziehungsweise den seriellen Ausgang des C 64 eingesteckt. Lediglich bei der Version für Epson-Drucker ist für die notwendige Stromversorgung des Interfaces zu sorgen. Dafür eignet sich am besten das gegen Aufpreis erhältliche Netzteil. Wer aber mit Lötkolben und Meßgeräten vertraut ist, wird in der Bedienungsanleitung

Hersteller (Typ) Merkmal	Görlitz Computerbau VCEI	Wiesemann (Typ 9200 für NEC)	Wiesemann (Typ 9200 für Epson)	Data Becker (VCI)	Eyssele	Kalawsky Ing. Büro	Bockstaller
Preis ca:	342,—	298,—	248,— ohne 298,— mit Netzteil	298,—	50,—	85,— + (Material und Kabel)	130,—
Art der Schnittstelle	Hardware	Hardware	Hardware	Hardware	Software	Software	Software
Form der Schnittstelle	Einbau in Epson-Drucker	extern in eigenem Gehäuse	extern in eigenem Gehäuse	Einbau in Epson-Drucker	Diskette und Kabel	EPROM und Kabel	EPROM und Kabel
Grafikzeichen	Ja	Ja	Nein	Ja	Ja	Nein	Nein
Reverse Zeichen	Ja	Ja	Nein	Ja	Ja	Nein	Nein
Listing Modus	Ja	Ja	Nein	Ja	Ja	Nein	Nein
Direkt Modus	Ja	Ja	Ja	Ja	Ja	Ja	Ja
Geräteadresse	4 einstellbar 0-15	4 einstellbar 5	4 einstellbar 5	4 einstellbar 5	4/16/17/18/19 mit besonderen Funktionen	4	4
Grafikfähigkeit	Ja	Ja	Nein	Ja	Nein	Nein	Nein
Hardcopy mit	Ja mit Simons Basic	Ja mit DB Supergrafik Ja, mit Simons B.	Nein	Ja mit DB Supergrafik Ja, mit Simons B.	Nein	Ja, von bel. Bild auf Tastendruck	Nein
Zeitbedarf für eine Hardcopy ca.	0:45 min	4:00		4:00		0:45 einf. 2:30 dopp.	
Druck mit: SM-Text	Ja	Ja	Ja	Ja	Nein	Ja	Nein
Textomat	Ja	Ja	Ja	Ja	Nein	Ja	Nein
Vizawrite	Ja	Ja	Ja	Ja	Ja	Ja	Ja
Wordpro 3+ Wordpro 1526	Ja	Ja	Ja	Ja	Nein	Nein	Nein
Druck mit CP/M	Ja	Ja	Ja	Ja	Nein	Nein	Nein

Bild 3. Die Vergleichstabelle der Testkandidaten

des Interfaces noch auf eine zweite Möglichkeit hingewiesen: die Entnahme der Versorgungsspannung von der Druckerplatine. Dieser Eingriff erfordert allerdings größte Vorsicht. Denn ein falscher Anschluß kann den »Tod« einer ganzen IC-Familie bedeuten.

Die Merkmale der Epson-Version sind rasch beschrieben, da sie sich lediglich auf die Ausgabe von Texten beschränken. Hierfür ist diese Schnittstelle mit Groß- und Kleinschriftmodus, sowie einigen Textformatierungsbefehlen ausgerüstet. Durch die Bauweise als Hardwareschnittstelle tauchten keine Probleme mit den zur Verfügung stehenden Textverarbeitungsprogrammen auf. Die Darstellung von Grafik- und Steuerzeichen ist leider nicht möglich. In diesem Bereich bietet das 9200 (NEC) die gleichen Funktionen wie das Data Becker VCI, kaum verwunderlich, wenn man weiß, wer der Hersteller des VCI ist: Wiesemann. Der auffallendste Unterschied vom 9200 NEC zum VCI ist die Möglichkeit, den Drucker in einen 6-Punkt-Modus zu schalten. Dadurch bleiben Programme, die für den 1515 Commodore-Drucker entwickelt wurden, weiterhin verwendbar.

Im wesentlichen gilt für das 9200 NEC-Interface das gleiche wie für das Data Becker-VCI. Lediglich auf den Einbau in den Drucker kann verzichet werden.

Die Epson-Version des 9200 eignet sich vor allem für jenen Computerbesitzer, die sich auf das problemlose Ausdrucken von Texten beschränken wollen. Allerdings liegt der Preis mit 248 ohne und 298 Mark mit Netzteil unverhältnismäßig hoch.

Der Außenseiter

Besonders gespannt waren wir auf das Abschneiden der Schnittstelle unseres Lesers H.Eyssele im Vergleich zu den professionellen. Wie die Verbindung zwischen dem Computer und dem Drucker hergestellt wird, braucht sicher nicht mehr erklärt werden, da dies genauestens in einem eigenen Bericht beschrieben ist. Ganz besonders gefallen haben uns die Möglichkeiten des Listingausdrucks. Die Umwandlung der Steuerzeichen in Klarschrift ist eine Funktion, die kein anderes Interface anbietet. Dadurch spart man sich beim Eintippen oder Korrigieren eines Listings das Nachschlagen der Steuercodes in der Vergleichstabelle. Die vorliegende Version der Schnittstelle funktionierte zwar nur in Verbindung mit Vizawrite, kann aber sicherlich ohne großen Aufwand an andere Programme angepaßt werden. Eine Hardcopy war leider noch nicht möglich. Hier geht der Aufruf an alle Programmierer, dies etwa als Teil unseres Programmierwettbewerbs »Programmbibliothek« zu ergänzen. (Die Redaktion wartet gespannt darauf).

Zum Preis von zirka 50 Mark (für das Kabel) bietet dieses Interface Leistungsmerkmale, wie sie eigentlich nur von professionellen Schnittstellen zu erwarten sind. Damit sicherte sich diese Lösung einen der vorderen Plätze der Bewertung.

Der Hardcopyspezialist

Die Kalawsky-Schnittstelle, deren Software nicht von Diskette, sondern durch ein Steckmodul in den C 64 eingeladen wird, überraschte durch eine komfortable Hardcopyroutine. Ein beliebiger Bildschirminhalt kann durch einfachen Tastendruck (CTRL-) auf den Drucker übertragen werden. Das Hardcopy benötigt etwa die gleiche Zeit wie das Görlitz-Interface mit Simons Basic. Ein Drücken der Tasten CTRL und x ermöglicht sogar ein großes Hardcopy auf die gesamte Blattbreite. Da manche Bilder als inverser Ausdruck besser aussehen, kann das Hardcopy auch in dieser Form ausgegeben werden. Natürlich ist auch die Textausgabe mit der Kalawsky Schnittstelle möglich. Dafür sollte man allerdings am besten ein Textverarbeitungsprogramm verwenden, da bei der direkten Ausgabe von Print-Zeilen im Groß/Kleinschriftmodus des C 64 Großbuchstaben als kursive Zeichen gedruckt werden. Dadurch leidet die Lesbarkeit doch stark. Mit 85 Mark für das Eprom, wo noch die Kosten für ein Verbindungskabel (zirka 45 Mark) hinzukommen, liegt der Preis sicher nicht in unerschwinglichen Höhen.

Für jeden, der sowohl Texte verarbeiten, als auch Hardcopys von beliebigen Bildern anfertigen möchte, ist dieses Interface ein nützliches Instrument.

Der äußeren Form des Kalawsky-Interface ähnlich präsentiert sich das Bockstaller-Interface. Die Bestandteile sind eine Epromplatine und ein Verbindungskabel. Diese Lösung macht sowohl Textausgaben von Basic aus, als auch mit dem Textverarbeitungsprogramm Vizawrite möglich. Der Groß/Kleinschriftmodus bleibt in jedem Fall erhalten. Für die Ausgabe von Grafiken und Hardcopys ist diese Schnittstelle allerdings weniger geeignet. Mit diesen Leistungen ist sie bei einem Preis von zirka 130 Mark zwar etwa gleich teuer wie das Kalawsky-Interface, besitzt aber nicht deren komfortable Hardcopyroutine.

Die Bewertung

Der Test hat gezeigt, daß jede Schnittstelle mit besonderen Fähigkeiten und Vorzügen aufwarten kann. An der Spitze des Testfeldes plazierten sich unserer Meinung nach die drei Hardwarelösungen von Görlitz, Data Becker und Wiesemann (NEC). Dabei belegte das Görlitz-Interface wegen seiner universellen Anwendbarkeit und dem vorbildlichen Aufbau der Platine den ersten Platz. Zwischen den beiden Interfaces von Data Becker und Wiesemann war der Unterschied nur gering. So wurde salomonisch beschlossen, diesen beiden Schnittstellen den zweiten Platz gemeinsam anzuerkennen. Für die Überraschung sorgte der Außenseiter: Der dritte Platz konnte an die Lösung unseres Lesers vergeben werden, und das lag nicht nur an dem unübertroffenen Preis-Leistungsverhältnis. Knapp geschlagen plazierte sich das Kalawsky-Interface auf dem vierten Platz. Mit seiner Hardcopyroutine auf Tastendruck und der Möglichkeit der Textverarbeitung bietet es zu einem annehmbaren Preis eine gute Leistung. Die Schlußlichter des Testes bildeten das Wiesemann (Epson) und das Bockstaller-Interface. Obwohl sich das 9200 (Epson) zur reinen Textverarbeitung hervorragend eignet, ist es mit einem Preis von zirka 298 Mark doch etwas teuer geraten. Das Bockstaller-Interface ist eine Lösung für alle, die ohne große finanzielle Ausgabe Texte ausdrucken wollen und sich die Zeit zum Laden der Software sparen wollen.

Sicherlich muß jeder potentielle Käufer eines Interfaces sich über seine eigenen Anwendungszwecke im Klaren sein. Auch spielt der zur Verfügung stehende Geldbetrag eine wesentliche Rolle. Die Entscheidung für eines der getesteten Interfaces ist deshalb immer eine individuelle, bei der nicht vergessen werden sollte, daß die Ansprüche an den Drucker mit der Zeit bestimmt nicht geringer werden.

(Arnd Wängler)

EXPANSIONS

Erweiterungen — das ist das Zauberwort, wenn man die durch Hardware oder Software ges

Ob man seinen Commodore 64 für die Meßdatenerfassung im Laboreinsatz umrüsten möchte oder ob man nur zusätzliche 16 KBytes RAM für seinen VC 20 benötigt — für fast jede denkbare Anwendung ist eine geeignete Erweiterung auf dem Markt. Das Angebot ist dabei äußerst vielfältig und selbst für den Fachmann kaum noch überschaubar, da eine Unzahl von Erweiterungen — für zum Teil sehr spezielle Zwecke — nur in kleinen Serien hergestellt und verkauft werden. Im 64'er wollen wir an dieser Stelle regelmäßig einen Überblick über Erweiterungen wie auch über interessante Neuentwicklungen für den Commodore 64 und den VC 20 geben.

(gk)

Produktbezeichnung	Preis in DM	geeignet für	Anbieter
Interfaces			
Centronics-Schnittstelle	130,00	VC20 C64	Bockstaller
Drucker-Spooler	280,00	VC20 C64	Bockstaller
IEEE-Schnittstelle	198,00	VC20 C64	Bockstaller
Schaltinterface 220 V	185,00	VC20 C64	Bockstaller
Centronics Eingang für VC20 C64	298,00	Drucker	Data Becker
Druckerinterface Centronics parallel	38,50	VC20 C64	Data Becker
IEC Bus Modul VC20	198,00	VC20	Data Becker
IEC Bus Modul C64	248,00	VC20 C64	Data Becker
Interpod IEEE-Universal-Interface	498,00	VC20 C64	Data Becker
V.24 Schnittstelle	128,00	VC20 C64	Data Becker
IEC Interface	249,00	VC20	Jeschke, Klaus
Recorder Interface	49,00	VC20 C64	Jeschke, Klaus
IEEE-Interface mit Basic 4.0 (Autostartmodul)	300,00	C64	Maier Datensys
Steckkarten/Module			
4-fach Steckplatine für Modulsteckplatz	175,00	VC20 C64	Bockstaller
Eprom-Platine für Modulsteckplatz	69.00	VC20 C64	Bockstaller
Input/Output Port-Modul 18 x 8 Bit	495,00	VC20 C64	Bockstaller
Input/Output Port-Modul 3 x 8 Bit	198,00	VC20 C64	Bockstaller
Modul Box + 8 KByte RAM	139,00	VC20	Data Becker
Modul Box VC 1020 bis 6 Module	389,00	VC20	Data Becker
Modul Box für 3 Steckplätze	89,00	VC20	Data Becker
Winkeladapter für 2 Module	99,00	VC20	Data Becker
Expansionsplatine (Bausatz)	99,00	C64	Hofacker GmbH
Externe Experimentierplatine	39,00	C64	Hofacker GmbH
Universelle Experimentierplatine	59,00	VC20	Hofacker GmbH
32 KByte RAM-Modul	179,00	VC20	Jeschke, Klaus
Steckadapter 3	99,00	VC20	Jeschke, Klaus
Steckbox 3	198,00	VC20	Jeschke, Klaus
Busplatine für 6 Module	198,00	VC20	KFC
Erweiterungsplatine für 3 Module	125,00	VC20	KFC
Erweiterungsplatine für 3 Steckplätze	125,00	C64	KFC
KFC-Super (Centronics, Toolkit, Fast-Tape u.a.)	198,00	C64	KFC
KFC-Super (Centronics, Toolkit, Fast-Tape u.a.)	150,00	VC20	KFC
KFC-Super Anschlußkabel für Centronics	60,00	VC20 C64	KFC
Erweiterungsplatine 5 Steckplätze	212,00	C64	Kalawsky
64 KByte RAM Modul	239,00	VC20	Roos electroni
Steckplatz für 2 Karten	69,00	C64	Roos electroni
Steckplatz für 2 Karten	69,00	VC20	Roos electroni
Steckplatz für 5 Karten	139,00	C64	Roos electroni
Steckplatz für 5 Karten	139,00	VC20	Roos electroni
64 KByte RAM Modul	338,00	VC20	Strie
Bus Platine, 6 Steckplätze, +3 KByte, +EPROM-Platz	198,00	VC20	Strie
80-Zeichen-Karten			
80 Zeichenkarte u. Centronics-Schnittstelle	198,00	C64	Bockstaller
80 Zeichenkarte u. Centronics-Schnittstelle	248,00	VC20	Bockstaller
80 Zeichen Modul Maxi	448,00	C64	Data Becker
80 Zeichen Modul Maxi	398,00	VC20	Data Becker
80 Zeichenkarte Modul	249,00	VC20	Jeschke, Klaus
80 Zeichenkarte	279,00	C64	Roos electroni
40/80 Zeichen Modul	348,00	VC20	Strie

EXPANSIONS

etzten Grenzen eines Computers überwinden will.

Messen und Steuern			
12 Bit A-D-Wandler	240,00	VC20 C64	Bockstaller
12 Bit D-A-Wandler	273,00	VC20 C64	Bockstaller
8 Bit 16-Kanal A-D	290,00	VC20 C64	Bockstaller
8 Bit A-D-Wandler	120,00	VC20 C64	Bockstaller
8 Bit D-A-Wandler	80.00	VC20 C64	Bockstaller
Quickfinger, steuert Joystick, an Controlport	49,00	VC20 C64	KFC
Software			
Austrocomp Basic Compiler	354,00	C64	Digimat
Dongle Programmschutz	138,00	C64	Digimat
Help Programmierhilfe	156,00	C64	Digimat
Help+ Programmierhilfe und Assembler	276,00	C64	Digimat
Basic Compiler	298,00	C64	Interface Age
Exbasic Level II Spracherweiterung	298,00	VC20 C64	Interface Age
Musiksynthesizer	98,00	C64	Interface Age
T.EX.AS Assembler-Entwicklungssystem	298,00	VC20 C64	Interface Age
Microsoft Multiplan	169,00	VC20 C64	Microdex GmbH
Grafik			
Grafik-ROM für Epson MX 80/MX 82	79,00	VC20 C64	Bockstaller
Grafik-ROM für Epson RX 80	145,00	VC20 C64	Bockstaller
Grafpad, hochaufl.Präzisionsgrafik Tablett	898,00	C64	Boston Computer
Lichtgriffel für Medium Resolution-Spiele	178,00	VC20 C64	Boston Computer
Proficolor 80 + Grafikbefehle (DM 88,- Disk)	298,00	C64	KFC

Anbieter/Herst.	Telefon	Straße	Plz/Ort
Bockstaller	07761-1808	Hadwigstr. 16	7867 Wehr-Öfflingen
Boston Computer	089-491073	Rosenheimer Str. 145a	8000 München 80
Data Becker	0211-3100-0	Merowingerstr.	4000 Düsseldorf 1
Digimat	AU-0222-542892	Arbeitergas. 48	A-1050 Wien
Hofacker GmbH	08024-7331	Tegernseerstr. 18	8150 Holzkirchen
Ing. Büro Kalawski	06150-2541	Fr. Ebert Str. 41	6108 Weiterstadt 1
Interface Age	089-5806702	Vohburger Str. 1	8000 München 21
KFC	06174-21953	Wiesenstr. 18	6240 Königstein 1
Klaus Jeschke	06198-7523	Im Birkenfeld 3	6233 Kelkheim
Maier Datensysteme	07721-70322	Gnädlingstr. 5	7730 VS-Villingen
Microdex GmbH	08152-1091	Mühlfelder Str. 2	8036 Herrsching/a.H.
Roos Elektronic	02821-28826	Kleiner Markt 7	4190 Kleve
Strie	04277-692	Kirchweg 5	2831 Schwaförden

EXPANSIONS

Der Data-Kassettenrecorder

In der Ausgabe 6/84 konnten wir in dem Artikel »Rund um die Datasette« wegen eines Gerätedefektes den Data-Kassettenrecorder von Nettetaler nicht berücksichtigen.

Der Recorder stellt mit dem Preis von 109 Mark eine preiswerte Alternative zur Original Datasette dar. Äußerlich ist dieses Gerät nicht besonders ansprechend. Im Gegensatz zur Datasette macht der Data-Kassettenrecorder einen zerbrechlichen Eindruck.

Im Testbetrieb, der für Geräte bei uns in der Redaktion zur Belastungsprobe wird, stellte sich der Recorder als zuverlässig heraus. Es traten keine Lade- und Speicherfehler auf. Auch bei den von uns benutzten Kassetten, die auf anderen Recordern oder der Orginal Datasette aufgenommen worden sind, gab es keine Probleme.

Nach unserer anfänglichen Enttäuschung über das defekte Testgerät stellte sich der Data-Kassettenrecorder als echte Alternative zur Original Datasette vor. Defekte an diesem Gerät sollen nach Auskunft von Nettetaler sehr selten sein: Die Rücklaufquote soll sich bei 12000 verkauften Geräten unter 0.5 % halten.

(rg)

Teleterm – Die Verbindung zum Modem



Um mit dem Commodore 64 über Akustikkoppler und Modem mit anderen Computern Daten austauschen zu können, benötigen Sie die entsprechende Software. Teleterm ist eine solche Treibersoftware.

Die RS-232-Schnittstelle

Hauptmenü von »Teleterm«

Um mit Ihrem C 64 in die Welt der Datenfernübertragung eintreten zu können, brauchen Sie, neben Ihrem Computer und einem Monitor (Fernseher), drei zusätzliche Werkzeuge. Als erstes wäre ein Modem oder ein Akustikkoppler für die Verbindung zum Telefon notwendig. Dann benötigen Sie ein Interface, um die Verbindung zwischen Akustikkoppler und Computer herzustellen. Das letzte, fast wichtigste Werkzeug ist die Treibersoftware. Erst dann können die Daten auch tatsächlich übertragen werden. Eines dieser Programme ist Teleterm von Software Express.

Am Anfang des Programms kann man die Sprache auswählen, mit der man arbeiten will. Zur Auswahl stehen Deutsch und Englisch.

Teleterm bietet über ein komfortables Menü vielseitige Möglichkeiten. Man kann zum Beispiel ein ständiges Protokoll auf der Diskette abspeichern. Dieses Programm lädt dabei alle Daten in einen Puffer, der dann, sobald er gefüllt ist, die Daten auf die Diskette schreibt. Der Schreibvorgang wird durch Änderung der Rahmenfarbe angezeigt. Diese Protokollierung ist oft sehr nützlich, kann aber auch zum Problem werden. Manche Mailboxen und Datenbanken haben einen Timeout. Timeout bedeutet: Wenn eine bestimmte Zeit kei-



Der Epson CX-21 Akustikkoppler

ne Kommunikation erfolgt, wird die Verbindung als beendet angesehen. Diese Regelung ist notwendig, da sonst ein falsch aufgelegter Telefonhörer ein System über Stunden blockieren könnte.

Wenn dieser Timeout sehr knapp eingestellt ist, kann das Abspeichern der Kommunikation zum Abbruch der Verbindung führen.

Die Menüpunkte sind im einzelnen:
— Daten empfangen mit Telefon
— Parameter ändern
— Dateien anzeigen
— Dateien ausdrucken
— Funktionstasten belegen
— Funktionstasten-Übersicht
— Vorbereiten von Briefen
— Dateien löschen
— Ende.

Bei diesem Menü fallen zwei Punkte besonders auf. Zum einen ist dies die Funktion »Parameter ändern«. Ruft man die Funktion »Parameter ändern« auf, dann können folgende Einstellungen verändert werden: die Übertragungsgeschwindigkeit (Baudrate), die Parität, die Wortlänge, die Anzahl der Stopbits, den Handshake-Betrieb und die Sendeart (Vollduplex/Halbduplex).

Die Bedeutung der oben erwähnten Begriffe entnehmen Sie bitte unserer Übersichtstabelle.

Der zweite auffällige Punkt ist die Möglichkeit, die Funktionstasten zu belegen. Besonders dieses Statement macht das Programm sehr komfortabel. Man kann die Funktionstasten zum Beispiel mit häufig vorkommenden Textstrings belegen und sich so viel Tipparbeit ersparen.

Der Testbetrieb

Teleterm stellte sich während des mehrerer Wochen dauernden Tests in der Redaktion als zuverlässiges Werkzeug heraus. Der Test wurde mit dem RS232-Interface desselben Anbieters wie Teleterm und dem Epson CX-21 Akustikkoppler durchgeführt. Das Handbuch wurde nach kurzer Zeit beiseite gelegt. So leicht ist Teleterm zu bedienen. Betreibt man den Akustikkoppler allerdings über eine ältere Nebenstellenanlage, kann es vorkommen, daß die Abschirmung der Telefonleitungen nicht ausreicht. Dies bewirkt, daß durch andere Leitungen falsche Zeichen eingelesen werden. Dieses Manko dürfte allerdings nur sehr schwer zu beseitigen sein und tritt nicht nur bei dieser Treibersoftware auf. Auf der Nebenstellenanlage unseres Verlages traten im Testbetrieb jedoch keine Probleme dieser Art auf.

Das Programm Teleterm kostet 119 Mark. Zu dem selben Preis wird von Software Express auch ein RS-232-Interface angeboten. Dieses Interface ist natürlich auch unabhängig von Teleterm verwendbar. (rg)

Electronic Mail — Die neue Form

Hauptanwendungsgebiet für Textverarbeitungsprogramme auf Personal- und Homecomputern ist das Schreiben von Briefen. Hier haben die Computer wesentliche Leistungs-, Zeit- und Kostenvorteile ermöglicht. Die Übermittlung der schriftlichen Mitteilungen erfolgt aber immer noch auf dem traditionellen Postweg mit langen Laufzeiten und hohen Kosten.

Immer deutlicher zeichnet sich ab, daß die Computer bald auch die Übermittlung von Briefen und Mitteilungen einschneidend verändern werden. Die großen Computerfirmen haben schon seit Jahren Systeme in Betrieb, mit denen sie dem teuren Briefverkehr ausweichen. Es handelt sich um Electronic-Mail-Systeme, die oft weltweit ausgelegt sind und die Mitarbeiter in die Lage versetzen, jederzeit und blitzschnell miteinander zu kommunizieren. Bildschirmterminals und Personal Computer werden über Modem oder Akustikkoppler ans Telefonnetz angeschlossen. Ebenfalls ans Telefonnetz angeschlossen ist ein Zentralcomputer, auf dem ein Mail-Box-Programm läuft. Dieses Programm richtet den Mitarbeitern elektronische Briefkästen und Editierplätze ein. Benutzerkennummern und Paßwörter ermöglichen es den Mitarbeitern, über ihre Terminals und PCs Mitteilungen abzurufen und zu senden, von jedem beliebigen kommunikationsfähigen Terminal oder PC. Die »Zustellung« beziehungsweise Verwaltung der Mitteilungen erfolgt im Zentralcomputer, der 24 Stunden am Tag ansprechbar ist.

Einige Computerfirmen wie Tandem haben ihr Electronic Mail System den Mitarbeitern über die rein betriebliche Anwendung hinaus geöffnet. Die Begeisterung darüber ist groß. Neben technischen Fragen und Problemlösungen werden betriebsintern auch Clubinformationen oder Gesuche und Angebote für zum Beispiel Gebrauchtwagen, Wohnungen und ähnliches allen Betriebsmitgliedern zugänglich gemacht.

Gegenwärtig sind Electronic-Mail-Systeme fast nur bei großen Computerfirmen im Einsatz. Terminals, PCs und Modems waren teuer. Entsprechende Zentralcomputer mit Mailbox-Programm ebenfalls. Der rasante technische Fortschritt im PC- und Homecomputerbereich hat diese Voraussetzungen im letzten Jahr einschneidend verändert. Preiswerte Personal und Homecomputer sind über oft bereits eingebaute V.24-Schnittstellen kommunikationsfähig und in großen Stückzahlen vorhanden. Neue, integrierte Modemchips haben Modems und Akustikkoppler in einen Preisbereich gebracht, in dem sie massenhaft Absatz finden können.

In den USA und England zeigt sich jetzt ein sehr starker Trend zur breiten Nutzung von Electronic Mail. Zunächst waren es vor allem Unternehmen, die bei Rechenzentren Kapazitäten für Electronic Mail anmieteten. Inzwischen gibt es auch mehrere Firmen, die Electronic Mail mit eindrucksvollen Erfolgen privaten Home- und Personal Computerbesitzern anbieten. Abrufsoftware und entsprechende Modems für die Home- und Personal Computer sind als preiswerte Paketangebote, oft inklusive eines Electronic-Mail-Abbonnements, in allen Computershops erhältlich. Da Mailbox-Software (für die Zentralstation) inzwischen auch für eine Reihe leistungsfähiger PCs preiswert verfügbar ist, gründen immer mehr Privatpersonen und Clubs sogenannte Bulletin Boards, kleinere Electronic-Mail-Systeme, die in der Regel auch als Informationsdatenbanken ausgelegt sind.

Electronic Mail wird sich auch in der Bundesrepublik durchsetzen. Dies dürfte schneller geschehen, als auf den ersten Blick zu erwarten ist. Mächtig dazu beitragen wird ein neuer Service, den die Bundespost mit ungewohnter Geschwindigkeit und Innovationsfreude demnächst einführt. Es handelt sich um die Telebox, einen bundesweiten Electronic-Mail-Service für alle über Akustikkoppler oder Modem kommunikationsfähigen Personal Computer, portable Computer, Terminals und natürlich auch Homecomputer.

In Mannheim wird ein Telebox-Zentralcomputer installiert, der über das normale Telefonnetz oder über die Datex-Dienste zugänglich ist. Geboten wird ein Electronic-Mail-System mit beachtlichen Leistungsmerkmalen:

Jeder Nutzer des Telebox-Systems erhält eine eigene Adresse, die ihn zusammen mit dem persönlichen Paßwort als berechtigten Nutzer des Systems ausweist und ihm dadurch das Eingeben und Auslesen von Mitteilungen im System ermöglicht.

— Mobilität bei ständiger Erreichbarkeit. Von jedem beliebigen Ort kann sich der Benutzer über die Datexnetze oder das Telefonnetz an das System anschalten, auch mit einem transportablen, akustisch gekoppelten Datenendgerät, das in der Aktentasche mitgeführt und an jedem beliebigen Telefon benutzt werden kann. Eingegangene Mitteilungen werden ausgelesen, eine Antwort beziehungsweise eine Mit-

der Postbeförderung

teilung abgesetzt oder eine abgelegte Mitteilung aus dem Speicher abgerufen.
— Einfaches Abspeichern und schnelles Wiederauffinden von Mitteilungen (elektronischer Aktenschrank). Der Benutzer kann sich Ablagefächer anlegen, die er beliebig strukturiert und benennt.
— Editieren und Formatieren von Texten. Ein umfangreicher Vorrat an Befehlen erleichtert dem Benutzer das Erstellen von Texten, die ebenfalls in wiederaufrufbare Dateien abgespeichert werden können.
— Erleichterungen beim Absenden und Lesen von Mitteilungen.
Mit der Angabe von Mehrfachadressen läßt sich dieselbe Mitteilung in einem Arbeitsgang an mehrere Empfänger absenden.
Adreß-Verteiler können vom Benutzer selbst angelegt werden.
Die Abfrage von Kopfzeilen erleichtert die Übersicht, wenn mehrere Mitteilungen eingegangen sind.
Das Weiterleiten von eingegangenen Mitteilungen an andere Telebox-Adressen mit oder ohne Zusetzen von Kommentaren erlaubt eine rasche Bearbeitung.
Das Beantworten von eingegangenen Mitteilungen ist in vereinfachtem Format möglich.
Die Nutzung der Leistungen von Telebox kann zu rascherer und effizienterer Information zwischen den Partnern führen. Damit ist außerdem Zeit- und Kostenersparnis verbunden.
— Schwarzes Brett: Hier können Informationen im System bereitgehalten werden, auf die entweder alle Benutzer oder bestimmte Gruppen von Benutzern zugreifen können. Dabei können Sparten mit beliebigen Namen eingerichtet werden, die vom Benutzer gezielt abgefragt werden. Informationen, die rasch einen größeren Empfängerkreis erreichen sollen, lassen sich auf diese Weise einfach zugänglich machen.
— Verzeichnisse: Sie erleichtern das Herausfinden der Adressen der Partner, denen man Mitteilungen zuleiten möchte. Es können Kurznamen und firmeninterne Bezeichnungen als Referenzadressen festgelegt werden, die vom System in die allgemeinen Adressen umgesetzt werden.

Während bei Bildschirmtext die Vorbereitung Jahre in Anspruch nahm und die Testphase mit beschränkter Teilnehmerzahl drei Jahre lief, startet die Telebox praktisch aus dem Stand. Im August bis September beginnt ein Teilnehmer-Test-Betrieb. Diese Phase wird subventioniert und ist für alle Interessenten offen.

Da preiswerte Akustikkoppler und preiswerte Kommunikationssoftware für Home- und Personal Computer zum Teil schon auf dem Markt sind oder nicht mehr lange auf sich warten lassen werden, muß die Telebox bei jedem Computerbesitzer Freude aufkommen lassen. Was allerdings bisher über die Telebox bekannt ist, trübt die Freude etwas. Zielgruppe der Bundespost

Die Telebox

sind hauptsächlich große und mittlere Unternehmen, denen die Telebox eine verbesserte und preiswertere Kommunikation zwischen Außendienst und Zentrale bringen soll. Bei den langen Laufzeiten und hohen Kosten des normalen Briefverkehrs und der Immobilität des Telex ist dieses Ziel nicht schwer zu erreichen. Mit einer nutzungszeitabhängigen Gebührenstruktur (vermutlich 0.30 Mark pro Minute) und einem monatlichen Mindestbetrag (vermutlich 80 Mark) ist der neue Service für die private Nutzung mit Home- oder Personal Computer zu teuer.

Trotz der für die private Nutzung unfreundlichen Gebührenstruktur sollten Besitzer von Home- und Personal Computern und speziell Besitzer des C 64 den neuen Service begrüßen. So wie die Einführung des PC durch die marktstarke IBM das Interesse und das Angebot an Personal Computern verbreiterte, wird die Telebox den Weg zu einer Vielzahl von Electronic-Mail-Systems und Bulletin Boards in der Bundesrepublik ebnen. Bereits jetzt bieten private Unternehmen wie Time Share, Digital Equipment, General Electric und andere Electronic Mail an, natürlich noch für Firmenkunden und zu Preisen, die ähnlich liegen, wie bei der Telebox. Es werden neue Unternehmen auftreten, welche preiswerte, auf große Teilnehmerzahlen ausgerichtete Electronic-Mail-Services der Vielzahl von Homecomputer- und Personal Computer-Besitzern anbieten werden. Technisch und kommerziell gesehen sind solche neuen Unternehmungen möglich und attraktiv. Hindernd wirkt sich gegenwärtig nur die Rechtsunsicherheit darüber aus, ob solche Services auf privater Basis zulässig sind.

Die Post wäre gut damit beraten, Wettbewerb zuzulassen. Einmal wird ihre Basis an Gebühreneinnahmen verbreitert, da die Nutzer solcher Services auf alle Fälle das Leitungsnetz der Post in Anspruch nehmen müssen. Die Nachrichtenvermittlung durch den Electronic-Mail-Service ist eine zusätzliche Dienstleistung, die sich innerhalb der Grundstücksgrenzen, sogar innerhalb des Computers des privaten Anbieters vollzieht. Das Monopol der Post auf Grundstücksgrenzen überschreitende Kommunikation wird gar nicht in Frage gestellt, und alle Erfahrungen der letzten Jahre sprechen dafür, daß private und kleinere Unternehmen bessere Chancen haben als die Post, innovative Mailbox-Software zu entwickeln und zu realisieren.

Im Bereich des Pakettransports konnten (nach einigen rechtlichen Schwierigkeiten) neue Unternehmen wie UPS, Deutscher Paketdienst und Ipec der Post Konkurrenz machen. Resultat ist ein wesentlich verbessertes Preis/Leistungsverhältnis im Paketdienst. Auch bei der Post, die im Wettbewerb jetzt ebenfalls neue Dienstleistungen anbietet.

Der Bereich der Telekommunikation ist für die zukünftige Entwicklung der Volkswirtschaft bestimmt nicht weniger wichtig als der des Pakettransports. Gerade bei der Telekommunikation hat die Bundesrepublik viel aufzuholen. Sollte man es sich ausgerechnet dann leisten, auf die Dynamik des Wettbewerbs und das heißt den Erfindungsgeist und die Initiative neuer Unternehmen zu verzichten? (Hersch Fischler)

Terminalprogramm für den C 64

Dieses Programm ermöglicht es Ihnen, über eine RS232-Schnittstelle und Akustikkoppler oder Modem mit anderen Computern in Verbindung zu treten. Der Austausch von Daten, Nachrichten und ganzen Programmen ist möglich — ebenso die Abfrage von Datenbanken. Wir haben dieses Programm mit der Software-Expreß-Schnittstelle und dem Epson CX-21-Akustikkoppler getestet und waren von Anfang an begeistert.

Das Terminalprogramm bietet folgende Möglichkeiten:

1. Die mit der Tastatur eingegebenen Zeichen werden über die RS232-Schnittstelle zur Gegenstelle gesandt.
2. Die von der Gegenstelle empfangenen Daten werden auf dem Bildschirm angezeigt.
3. Die von der Gegenstelle empfangenen Daten können zur weiteren Bearbeitung in den »Terminal-Speicher« geschrieben werden.
4. Diese gespeicherten Daten können dann wieder ausgesandt, auf den Bildschirm oder Drucker ausgegeben, beziehungsweise auf Diskette gespeichert werden.
5. Dateien können von der Diskette, zur weiteren Bearbeitung, in den »Terminal-Speicher« geladen werden.
6. Empfangene und im »Terminal-Speicher« befindliche Programme (als ASCII-Datei gespeichert) können über eine »Wandler«-Software aufbereitet werden. Hierbei besteht eine Kontroll- und Editiermöglichkeit der einzelnen Programmzeilen.
7. Die Übertragungsparameter sind dem Standard entsprechend eingestellt. Durch Änderung der beiden CHR$-Befehle in den »OPEN«-Anweisungen können andere Parameter eingestellt werden. Die entsprechenden »OPEN«-Anweisungen befinden sich in Basic-Zeilennummer 10 + 50. Die Werte für das Kontroll- und Befehlsregister (1. und 2. CHR$ Befehl) sind im Commodore Programmier-Handbuch aufgeführt.

Der »Terminal-Speicher« belegt den Adressenbereich Hex 5000-9FFF.

Der Basic-Speicher endet bei Hex 3FFF.

Das Terminalprogramm wird mit Load »terminal«, 8

```
10 PRINT"■  EINEN MOMENT BITTE "
20 FORX=16384TO16995:READY:DS=DS+Y:POKEX
,Y:NEXT
30 IFDS<>68997THENPRINT"FEHLER IN DATAS:
 SOLL=68997 IST="DS:STOP
40 POKE51,0:POKE52,64:POKE55,0:POKE56,64
:CLR:F=0:GOTO130
50 POKE53280,5:POKE53281;5 :PRINT"■"
60 OPEN2,2,0,CHR$(6+32      )+CHR$(32+64+1
28):GET#2,A$
70 PRINTCHR$(14):PRINT"■":PRINT"        DA
TENBANKDIALOG MIT DEM C64"
80 SYS16394
90 CLOSE2:  POKE53280,5:POKE53281,5:PRIN
T"■":F=1:  GOTO130
100 OPEN2,2,0,CHR$(6+32     )+CHR$(32+64+
128):GET#2,A$:PRINT"■":PRINT" NEUSTART !
"
110 SYS16416
120 GOTO90
130 PRINTCHR$(14):PRINT"■QQ■■■■■■■■■■■■■■
MENUE"
140 PRINT"QQ■■■■1 = BEDEUTUNG DER FUNKTI
ONSTASTEN"
150 IFF=0THEN170
160 PRINT"Q■■■■2 = ZURUECK INS TERMINALP
ROGRAMM":GOTO180
170 PRINT"Q■■■■2 = TERMINAL PROGRAMM  ST
ARTEN":GOTO180
180 PRINT"Q■■■■3 = EMPFANGENE UND GESPEI
CHERTE            DATEN DRUCKEN
190 PRINT"Q■■■■4 = EMPFANGENE UND GESPEI
CHERTE            PROGRAMME AUFBERE";
200 PRINT"ITEN"
210 PRINT"Q■■■■5 = EMPFANGENE UND GESPEI
CHERTE            DATEN AUF DISK. ";
220 PRINT"SPEICHERN"
230 PRINT"Q■■■■6 = DATEN VON DISK. IN DE
N SPEICHER        SCHREIBEN        "
240 PRINT"Q■■■ BITTE WAEHLEN 1-6 "
250 GETA$:IFA$=""THEN250
260 IFA$="1"THEN340
270 IFA$="2"ANDF=1THEN100
280 IFA$="2"THENF=1:GOTO50
290 IFA$="3"THEN520
300 IFA$="4"THEN570
310 IFA$="5"THEN650
320 IFA$="6"THEN730
330 GOTO250
```

geladen und anschließend mit Run gestartet. Nach einer kurzen Wartezeit, während der Maschinenspracheteil angelegt wird, erscheint das Menü auf dem Bildschirm.

Über das Menü und mit Hilfe der Funktionstasten werden alle Programmabläufe gesteuert.

Menüpunkt 1 erklärt die Bedeutung der Funktionstasten F1 bis F8, der ←Taste und der RUN/Stop Taste.

Funktionstaste F1:

Durch Betätigung werden empfangene Daten als ASCII-Datei in den »Terminal-Speicher« geschrieben. Ein empfangenes Hex 02-Zeichen (Start of Text) bewirkt ebenfalls ein Abspeichern. Zur Erinnerung ändert sich die Rahmenfarbe des Bildschirmes in Braun.

Funktionstaste F3:

Die Betätigung beziehungsweise ein empfangenes Hex 03-Zeichen (End of Text) beendet das Abspeichern. Die Bildschirm-Rahmenfarbe wird wieder grün.

Funktionstaste F5:

Mit dieser Taste wird an den Anfang des »Terminal-Speichers« zurückgesetzt und die erste Zeile (bis zum Return-Zeichen) auf den Bildschirm übertragen.

Funktionstaste F7:

Ein wiederholtes Drücken dieser Taste führt schrittweise bis ans Ende des Textes. Das Textende ist durch + + + gekennzeichnet. Ein Speicherplatzende wird ebenfalls durch + + + angezeigt.

Funktionstaste F2:

Mit dieser Taste wird ein Vorschreiben in den »Terminal-Speicher« eingeleitet. Die Zeichen werden gleichzeitig auf dem Bildschirm angezeigt. Hierbei sind die Cursor-Tasten außer Funktion. Falsch eingegebene Zeichen können mit der Del-Taste gelöscht werden.

Funktionstaste F4:

Diese Taste beendet das Vorschreiben.

Funktionstaste F6:

Durch Betätigung dieser Taste wird wieder an den Anfang des »Terminal-Speichers« zurückgesetzt. Bis zum ersten Return-Zeichen werden die Daten über die RS232-Schnittstelle ausgegeben und gleichzeitig auf dem Bildschirm angezeigt.

Funktionstaste F8:

Ein wiederholtes Drücken dieser Taste setzt den Sendebetrieb schrittweise bis zum Dateiende fort. Das Ende wird wieder durch + + + angezeigt.

Die ←Taste

Mit dieser Taste wird der Rücksprung in das Menü eingeleitet.

Die Run/Stop Taste

Das Terminalprogramm kann mit Run/Stop unterbrochen werden.

```
340 PRINT"BEDEUTUNG DER FUNKTIONS
TASTEN"
350 PRINT"F1 = EMPANGENE DATEN SPE
ICHERN"
360 PRINT"F3 = ABSPEICHERN BEENDEN
"
370 PRINT"F5 = GESPEICHERTE DATEN A
NSEHEN           ( ANFANG )"
380 PRINT"F7 = GESPEICHERTE DATEN A
NSEHEN           ( WEITER )"
390 PRINT"F2 = TEXT VORSCHREIBEN "
400 PRINT"F4 = VORSCHREIBEN BEENDEN
"
410 PRINT"F6 = GESPEICHERTE DATEN A
USSENDEN         ( ANFANG )"
420 PRINT"F8 = GESPEICHERTE DATEN A
USSENDEN         ( WEITER )"
430 PRINT"WEITER MIT RETURN"
440 GETA$:IFA$=""THEN440
450 PRINT"":PRINT"   DIE _ TASTE BEWIRK
T EINEN RUECK-       SPRUNG INS MENUE"
460 PRINT:
470 PRINT"   MIT RUN/STOP KANN DAS PROGR
AMM      ABGEBROCHEN WERDEN":PRINT
480 PRINT"   BEIM VORSCHREIBEN KANN MIT
DER      DEL-TASTE GELOESCHT WERDEN,
490 PRINT"   WEITER MIT RETURN"
500 GETA$:IFA$=""THEN500
510 GOTO130
520 OPEN1,4,7:REM DRUCKER VC1526
530 E=PEEK(16387)+PEEK(16388)*256
540 FORI=20480TOE-1
550 PRINT#1,CHR$(PEEK(I));:NEXT
560 CLOSE1:GOTO130
570 PRINT"AUTOMATISCHES WANDELN O
DER             MANUELLES WANDELN ?"
580 PRINT"BITTE WAEHLEN   ( A ODER
M )"
590 PRINT"MIT RETURN KOMMEN SIE ZU
RUECK           IN DAS MENUE"
600 GETA$:IFA$=""THEN600
610 IFA$="A"THENLOAD"WANDLER 2",8:RUN
620 IFA$="M"THENLOAD"WANDLER",8:RUN
630 IFA$=CHR$(13)THEN130
640 GOTO600
650 PRINT"":INPUT"DATEINAME";D$:P
RINT""
660 PRINT"DATEI: ";D$;" WIRD AUF
                DISKETTE GESPEICHERT"
670 CLOSE2:OPEN1,8,2,D$+",S,W"
680 E=PEEK(16387)+PEEK(16388)*256
690 FORI=20480TOE-1
700 PRINT#1,CHR$(PEEK(I));:NEXT
710 PRINT"":PRINT"DATEN SIND AUF
DISKETTE GESPEICHERT":FORI=1TO2000:NEXT
720 CLOSE1:GOTO130
730 PRINT"":INPUT"DATEINAME";D$:P
RINT""
740 OPEN1,8,2,D$+",S,R":I=20480
750 PRINT"DATEI: ";D$;" WIRD VON
                DISKETTE GELADEN"
760 GET#1,A$
770 POKEI,(ASC(A$)):I=I+1
780 IFST=64THENCLOSE1:GOTO810
790 IFST<>0THENCLOSE1:PRINT"DISKSTA
TUS ";ST;"!":FOR I=1TO5000:NEXT:GOTO130
800 GOTO760
810 POKE16387,(I-INT(I/256)*256):POKE163
88,(I/256):GOTO130
820 DATA0,0,0,0,0,0,0,0,0,0,169,0,141,1,
64,141,3,64,141,4,64,133,251,169
830 DATA80,133,252,169,194,141,5,64,32,2
28,255,201,0,240,85,201,20,240,3
840 DATA76,49,64,169,8,234,201,3,240,126
,201,133,240,125,201,134,240,124
850 DATA201,135,240,123,201,136,240,122,
201,137,240,121,201,138,240,120,201
860 DATA139,240,119,201,140,240,118,201,
95,240,117,201,65,144,23,176,0,201
870 DATA96,176,3,76,109,64,201,192,176,8
,76,116,64,105,32,76,116,64,233,128
880 DATA32,90,66,234,234,234,234,234,32,
134,240,201,0,240,157,201,13,240
890 DATA17,201,2,240,43,201,3,240,42,201
,32,144,233,201,128,176,229,234,201
900 DATA65,144,16,176,1,234,201,96,176,7
,24,105,128,76,172,64,234,233,32
910 DATA72,32,210,255,76,220,64,76,88,66
,76,212,64,76,54,65,76,190,65,76
920 DATA196,65,76,65,65,76,54,65,76,2,66
,76,11,66,76,51,65,0,0,0,169,1,141
930 DATA1,64,76,32,64,169,1,205,1,64,240
,12,104,24,72,169,245,141,32,208
940 DATA104,76,32,64,104,72,169,249,141,
32,208,104,32,0,65,76,32,64,0,0,0
950 DATA160,0,145,251,230,251,208,2,230,
252,24,165,251,141,3,64,201,255,240
960 DATA7,165,252,141,4,64,24,96,165,252
```

Menüpunkt 2 startet das eigentliche Terminalprogramm. Zum Datenaustausch ist hinzuzufügen, daß einige Steuerzeichen ausgefiltert werden, um ein einwandfreies Arbeiten mit anderen Datenbanken zu ermöglichen. Weiterhin wird eine Code-Wandlung zwischen CPM und ASCII durchgeführt. Beim Betätigen der Del-Taste wird ein Backspace (ein Zeichen zurück) zur Gegenstelle gesandt. Da die Cursor-Steuertasten ihre Funktion verloren haben, wurde auf die Anzeige des Cursors verzichtet.

Menüpunkt 3 bewirkt die Ausgabe der im »Terminal-Speicher« befindlichen Daten auf einen angeschlossenen Drucker

Menüpunkt 4 verläßt das Programm und lädt das »Wandler-Programm«. Mit diesem »Wandler« ist es möglich, ein empfangenes und im »Terminal-Speicher« abgelegtes Programm (als ASCII-Datei) in ein lauffähiges Programm zu wandeln und in den Basic-Speicher zu schreiben. Es kann dann benutzt oder auf Diskette gespeichert werden. Das »Wandler-Programm« startet selbstätig und auf dem Bildschirm erscheint: ready. »......Erste Programmzeile......« run 60020. Der Cursor blinkt in der ersten Programmzeile. Sind in dieser Zeile keine Fehler vorhanden, so kann diese durch Betätigen der Return-Taste in den Basic-Arbeitsspeicher übernommen werden. Jetzt steht der Cursor in der Zeile »run 60020«. Ein nochmaliges Drücken der Return-Taste bringt die nächste Programmzeile auf den Bildschirm. In gleicher Weise kann bis zum Programmende fortgefahren werden. Ist die Änderung einer Programmzeile erforderlich, so kann dieses mit den üblichen Editiermöglichkeiten geschehen. Soll eine Zeile nicht übernommen werden, so wird der Cursor manuell in die Zeile »run 60020« gesteuert und mit)Return(weitergearbeitet. Zu beachten ist, daß die Programmzeile nicht länger als 80 Zeichen ist. Der Menüpunkt 4 bietet zusätzlich eine Auswahlmöglichkeit zwischen »automatischer Wandlung« und »manueller Wandlung«. Die »manuelle Wandlung« wurde beschrieben. Bei der »automatischen Wandlung« läuft das Programm selbstätig ab. Wird aber eine fehlerhafte Zeile erkannt, so bricht das Programm ab. Nach der Editierung kann mit »run 60020« weitergearbeitet werden.

Menüpunkt 5 speichert alle im »Terminalspeicher« befindlichen Daten als sequentielle Datei auf eine Diskette. Vor dem Abspeichern muß der Datei-Name eingegeben werden.

Menüpunkt 6 lädt eine sequentielle Datei von der Diskette in den »Terminal-Speicher«.

```
,141,4,64,201,159,240,2,24,96,169
970 DATA42,32,210,255,32,210,255,32,210,
255,198,251,96,0,0,169,0,141,1,64
980 DATA76,32,64,0,0,0,169,1,141,1,64,32
,228,255,201,0,240,249,201,13,240
990 DATA24,201,20,240,20,201,32,144,237,
201,145,240,233,201,157,240,229,201
1000 DATA147,240,225,201,138,240,28,32,2
10,255,201,20,240,3,76,127,65,198
1010 DATA251,208,2,198,252,24,76,70,65,0
,0,32,0,65,76,70,65,76,54,65,0,0,0
1020 DATA0,0,169,0,133,251,169,80,133,25
2,165,252,205,4,64,240,3,76,174,65
1030 DATA165,251,205,3,64,240,3,76,174,6
5,24,76,215,65,0,160,0,177,251,230
1040 DATA251,208,2,230,252,24,234,234,23
4,234,96,32,141,65,32,210,255,32,149
1050 DATA65,32,210,255,201,13,240,3,76,1
96,65,76,32,64,0,0,0,32,38,65,169
1060 DATA13,96,0,0,0,201,65,144,23,176,0
,201,96,176,3,76,244,65,201,192,176
1070 DATA8,76,251,65,105,32,76,251,65,23
3,128,32,90,66,96,0,0,0,32,141,65
1080 DATA32,210,255,32,224,65,32,149,65,
32,210,255,32,224,65,201,13,240,185
1090 DATA76,11,66,76,32,64,0,201,65,144,
23,176,0,201,96,176,3,76,51,66,201
1100 DATA192,176,8,76,58,66,105,32,76,62
,66,233,128,32,210,255,96,233,31,76
1110 DATA58,66,0,32,141,65,32,31,66,32,1
49,65,32,31,66,201,13,240,3,76,74
1120 DATA66,96,96,0,133,158,32,8,242,96,
96,0,0,0

READY.
```

»Terminalprogramm« (Ende)

Soll ein Basic-Programm zur Gegenstelle gesandt werden, so sind einige Besonderheiten zu beachten.

1. Das Programm sollte nur »reine ASCII-Daten« erhalten. Bildschirm-Steuerzeichen sowie Grafikzeichen werden nicht übertragen.
2. Das Programm muß als sequentielle Datei vorliegen und nicht wie sonst üblich als Programm-Datei.

Die Wandlung einer Programm-Datei in eine sequentielle Datei wird folgendermaßen durchgeführt.

1. Das Programm wird wie üblich mit LOAD »NAME«, 8 in den Basic-Speicher des C 64 geladen.
2. Im Direktmodus wird folgende Zeile eingegeben: OPEN 1,8,2,»Name,S,W«:CMD1:LIST

Jetzt wird das Programm als sequentielle Datei auf die Diskette geschrieben und kann später vom Terminalprogramm aufgerufen werden. Anschließend wird mit CLOSE 1 das eröffnete File geschlossen.

(Manfred Wyrwas)

In der nächsten Ausgabe das entsprechende Treiber-Programm für den VC 20.



Begriffe aus der DFÜ

Wenn man über Datenfernübertragung spricht, tauchen oft Begriffe auf, deren Bedeutung nicht jedem klar ist. Deswegen werden hier einige der wichtigsten Begriffe erläutert.

Modem: Das Modem stellt die Verbindung zwischen Ihrem Computer und dem Telefonnetz her. Es wird von der Post direkt an das Telefonnetz angeschlossen. Das Modem wandelt die Ausgabesignale des Computers in elektrische Signale um, die über das Telefonnetz übertragen werden können.

Akustikkoppler: Der Akustikkoppler hat dieselben Funktionen wie das Modem. Er wird aber nicht wie das Modem direkt an das Telefonnetz angeschlossen. Der Akustikkoppler verbindet den Computer mit dem Hörer eines herkömmlichen Telefonapparates. Der Akustikkoppler sendet akustische Signale, die vom Telefonapparat in elektrische Impulse umgewandelt werden, und empfängt akustische Signale, die er für den Computer in elektrische Impulse umwandelt.

Handshaking: Handshaking bedeutet die Steuerung der Datenübertragung durch Stopp-, Sende- und Empfangssignale. Die Datenübertragung wird erst dann fortgesetzt, wenn der korrekte Empfang bestätigt wurde.

Vollduplex: Bei dieser Art der Datenübertragung sendet der Computer, an den man die eigenen Daten übertragen hat, diese Daten zurück, damit der eigene Computer sie auf korrekte Übermittlung überprüfen kann. Stimmen diese Daten überein, so wird mit der Übertragung fortgefahren. Ist die Übereinstimmung nicht gegeben, so werden diese Daten erneut gesendet.

Halbduplex: Im Halbduplex-Betrieb sendet der empfangende Computer nur eine Bestätigung des Empfangs.

Baud-Rate: Mit diesem Begriff wird die maximale Übertragungsgeschwindigkeit bezeichnet. Die Baud-Rate wird in BPS (Bits pro Sekunde) angegeben.

Paritätsbit: Die Paritätskontrolle ist ein Hilfsmittel zur Datensicherung. Dabei wird einer einheitlich langen Folge von Informationsbits ein zusätzliches Bit hinzugefügt. Der Wert dieses Bits wird nach Vereinbarung so gewählt, daß mit ihm die Anzahl der Bits mit dem Wert binär Eins über die gesamte Einheit hinweg stets einen geraden oder stets einen ungeraden Wert erreicht. Es wird dann von gerader oder ungerader Parität gesprochen.

Start/Stop-Bits: Bei jeder Start-Stop-Übertragung wird jedem n-ten Bit jeweils ein Bit voran- und nachgesetzt. Das vorangesetzte Bit wird Start-Bit genannt, das nachgesetzte wird als Stop-Bit bezeichnet.

Treibersoftware: Mit Treibersoftware wird das Programm bezeichnet, das die Kommunikation des Computers mit dem Modem steuert. Komfortable Versionen der Treibersoftware haben Zusatzfunktionen, die zum Speichern oder Drucken der gesendeten und empfangenen Daten dienen.

Mailbox: Bei einer Mailbox handelt es sich um eine Datenbank, in die Nachrichten geschrieben oder aus der Nachrichten gelesen werden können.

Answer/Originate: Wenn Sie mit einem Netz Verbindung aufnehmen, rufen Sie das System an. Da der Ruf von Ihnen ausgeht, muß Ihr Modem auf »Originate« gesetzt werden. Dadurch wird das »Einführungsprotokoll« in Gang gesetzt. Wenn Sie mit einem anderen Computer kommunizieren, muß einer im »Originate«- und der andere im »Answer«-Modus sein. Wenn die Verbindung aufgebaut ist, ist es nicht mehr von Bedeutung, welcher von beiden in welchem Modus ist.

(rg)

Wie bedient man eine Mailbox?

Einen Akustikkoppler kaufen ist nicht schwierig. Doch wie geht es weiter? Wie man in eine Mailbox einsteigt und andere Fragen werden in diesem Bericht beantwortet.

Vorweg die rechtliche Frage: Die Benutzung eines Akustikkopplers, um in eine Mailbox einzusteigen, ist völlig legal. Es wird sogar erwartet. Denn genau zu diesem Zweck wurden Mailboxen geschaffen. Eine Mailbox ist im Prinzip nichts anderes als eine Datenbank oder eine Datei, in die Nachrichten eingegeben und abgerufen werden können. Jeder Besitzer eines Akustikkopplers ist dazu in der Lage.

Welche Geräte sind notwendig?

Die zur Kommunikation notwendigen Geräte sind schnell zusammengestellt. Neben dem C 64/VC 20 benötigen Sie eine V.24-Schnittstelle. Sie wird in den User-Port des Computers gesteckt. Ein geeignetes Kabel verbindet den Akustikkoppler mit dem Interface. Das wäre alles. Jetzt fehlt nur noch die Software. Eine Möglichkeit ist das in diesem Heft abgedruckte Programm. Aber es gibt schon mehrere Programme, die diese Aufgabe erfüllen. Jeder Computertyp braucht seine auf ihn abgestimmte Software. Wenn Sie diese Geräte und das Programm haben, kann es losgehen.

Kein Anschluß ohne Nummer

Zwei Probleme müssen noch bewältigt werden: In der Nähe des Computers muß ein Telefonanschluß verfügbar sein. Ohne Telefon geht es nun mal nicht. Doch was nützt die beste Bedienungsanleitung des Akustikkopplers, wenn sie keine Rufnummer enthält, unter der eine Mailbox erreichbar ist. Aber ab sofort ist auch diese Schwierigkeit behoben. Ich werde Ihnen noch einige Telefonnummern geben, die Sie einmal ausprobieren sollten.

Der erste Kontakt

Anhand eines Beispiels möchte ich mit Ihnen eine Mailbox anwählen. Ich habe ein geeignetes Programm in den Computer geladen. Nach dem Starten steht mir ein Menü zur Verfügung, aus dem man eine Anzahl von Funktionen wählen kann. Ich wähle die Funktion »Daten laden«. Sie stellt die Aufnahmebereitschaft her. Die jetzt folgende Frage »Sollen die Daten auf Diskette gespeichert werden?« beantworte ich mit »Ja«, denn ich möchte mir nachher alles ausdrucken lassen. Und

```
***** KEINE MITTEILUNG FUER SIE *****

H A U P T M E N U E

1   D E C A T E S  ( Anschrift & Informationen )
2   M A I L B O X  ( Informations-Austausch )
3   S O F T B O X  ( Programm-Austausch )
4   T E L E B O X  ( Mailbox-Nummern-Liste )
8   F U N D B O X  ( Hardware / Software )
9   I N F O D A T  ( Nicht-Oeffentliche Datenbank )

Ihre EINGABE (1,2,3,4,8,9)  ==>  4

4   T E L E B O X  ( Mailbox-Nummern-Liste )

1   MAILBOXEN in DEUTSCHLAND   ( Telefon )
2   MAILBOXEN in DEUTSCHLAND   ( Datex P )
3   MAILBOXEN im AUSSLAND      ( Telefon )
4   MAILBOXEN im AUSSLAND      ( Datex P )

9   Zum HAUPTMENUE
Ihre EINGABE (1-4,9)  ==> 1

0211   / 593453EPSON
02161  / 200928SYMIC
02202  / 50033COMMODORE MAILBOX
02202  / 5008COMPUTER CENTR.
0231   / 650786CBBS
0231   / 7552541  IBM 370 DORTMUND
                 USEID: FINIX15    PASSWORD: THEOBALD
ENTRY UEBER: LOGON
0241   / 81081TH AACHEN

030    / 3055060BERLINER - MAILBOX
030    / 314730UNI BERLIN

040    / 4123309BUNI HAMBURG   ( 20 - 6h )
040    / 6523486M.C.S.
0431   / 8804556UNI KIEL

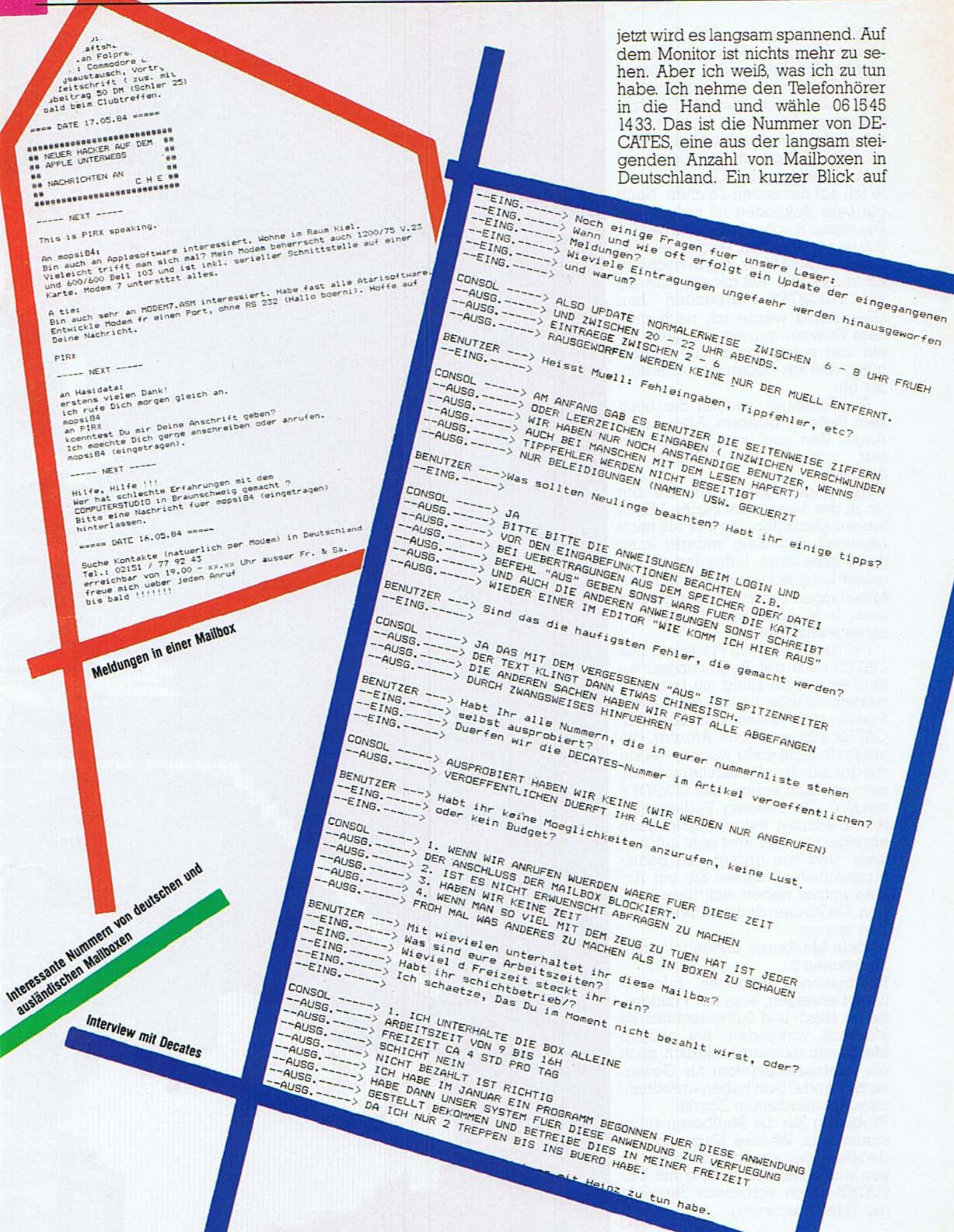
0611   / 816787TECOS

089    / 222066GRAPHTON RECHENZENTRUM
089    / 280310CYBER .
089    / 32095264LOGIN 11,3  help
089    / 596422FRANZIS ( TEDAS )
089    / 598423FRANZIS ( TEDAS )
-----------------------------------------------
4   T E L E B O X  ( Mailbox-Nummern-Liste )
-----------------------------------------------
Ihre EINGABE (1-4,9)  ==> 4

Address             MnemonicEstablishment

2062221006BBDABrussels DEC A (Belgium)
208034020258CNUSCCNUSC Montpelier
208078020169O1INRIAInstitute Nucleare Reserche .........
208091000309*DCISIFMSTCISI1IBM (TSO)
208091000309*DCISIFMSTCISI1IBM (TSO)
208091000519*DCISIFMSTCISI2IBM - TSO
208091000270*DCISIFMSTCISI3IBM - TSO
208091010320CJRCE
208091040047SACLAYSaclay - France
2223077*DQUESTD5ESAESA
2223078*DQUESTD5ESA2ESA
2283101*DNDATASTARData-Star, Switzerland
22846811405CERNCERN
```

jetzt wird es langsam spannend. Auf dem Monitor ist nichts mehr zu sehen. Aber ich weiß, was ich zu tun habe. Ich nehme den Telefonhörer in die Hand und wähle 06 15 45 14 33. Das ist die Nummer von DECATES, eine aus der langsam steigenden Anzahl von Mailboxen in Deutschland. Ein kurzer Blick auf



```
ji.
aftsh.
.an Folpre.
: Commodore u
ysaustausch, Vortr,
Zeitschrift ( zus. mit
beitrag 50 DM (Schler 25)
bald beim Clubtreffen.

==== DATE 17.05.84 =====

##########################
## NEUER HACKER AUF DEM  ##
## APPLE UNTERWEGS       ##
##                       ##
## NACHRICHTEN AN  C H E ##
##
##########################

----- NEXT -----

This is PIRX speaking.

An mopsi84:
Bin auch an Applesoftware interessiert. Wohne im Raum Kiel.
Vieleicht trifft man sich mal? Mein Modem beherrscht auch 1200/75 V.23
und 600/600 Bell 103 und ist inkl. serieller Schnittstelle auf einer
Karte. Modem 7 unterstzt alles.

A tim:
Bin auch sehr an MODEM7.ASM interessiert. Habe fast alle Atarisoftware.
Entwickle Modem fr einen Port, ohne RS 232 (Hallo boerni). Hoffe auf
Deine Nachricht.

PIRX

----- NEXT -----

an Hasidata:
erstens vielen Dank!
ich rufe Dich morgen gleich an.
mopsi84
an PIRX
koenntest Du mir Deine Anschrift geben?
Ich moechte Dich gerne anschreiben oder anrufen.
mopsi84 (eingetragen).

----- NEXT -----

Hilfe, Hilfe !!!
Wer hat schlechte Erfahrungen mit dem
COMPUTERSTUDIO in Braunschweig gemacht ?
Bitte eine Nachricht fuer mopsi84 (eingetragen)
hinterlassen.

===== DATE 16.05.84 =====

Suche Kontakte (natuerlich per Modem) in Deutschland
Tel.: 02151 / 77 92 43
erreichbar von 19.00 - xx.xx Uhr ausser Fr. & Sa.
freue mich ueber jeden Anruf
bis bald !!!!!!!
```

Meldungen in einer Mailbox

```
--EING.------>
--EING.------> Noch einige Fragen fuer unsere Leser:
--EING.------> Wann und wie oft erfolgt ein Update der eingegangenen
--EING.------> Meldungen?
--EING.------> Wieviele Eintragungen ungefaehr werden hinausgeworfen
--EING.------> und warum?

CONSOL ------> ALSO UPDATE NORMALERWEISE ZWISCHEN 6 - 8 UHR FRUEH
--AUSG.------> UND ZWISCHEN 20 - 22 UHR ABENDS.
--AUSG.------> EINTRAEGE ZWISCHEN 2 - 6
--AUSG.------> RAUSGEWORFEN WERDEN KEINE NUR DER MUELL ENTFERNT.

BENUTZER ---> Heisst Muell: Fehleingaben, Tippfehler, etc?
--EING.------>

CONSOL ------> AM ANFANG GAB ES BENUTZER DIE SEITENWEISE ZIFFERN
--AUSG.------> ODER LEERZEICHEN EINGABEN ( INZWICHEN VERSCHWUNDEN
--AUSG.------> WIR HABEN NUR NOCH ANSTAENDIGE BENUTZER, WENNS
--AUSG.------> AUCH BEI MANSCHEN MIT DEM LESEN HAPERT)
--AUSG.------> TIPPFEHLER WERDEN NICHT BESEITIGT
--AUSG.------> NUR BELEIDIGUNGEN (NAMEN) USW. GEKUERZT

BENUTZER --->Was sollten Neulinge beachten? Habt ihr einige tipps?
--EING.------>

CONSOL ------> JA
--AUSG.------> BITTE BITTE DIE ANWEISUNGEN BEIM LOGIN UND
--AUSG.------> VOR DEN EINGABEFUNKTIONEN BEACHTEN Z.B.
--AUSG.------> BEI UEBERTRAGUNGEN AUS DEM SPEICHER ODER DATEI
--AUSG.------> BEFEHL "AUS" GEBEN SONST WARS FUER DIE KATZ
--AUSG.------> UND AUCH DIE ANDEREN ANWEISUNGEN SONST SCHREIBT
--AUSG.------> WIEDER EINER IM EDITOR "WIE KOMM ICH HIER RAUS"

BENUTZER ---> Sind das die haufigsten Fehler, die gemacht werden?
--EING.------>

CONSOL ------> JA DAS MIT DEM VERGESSENEN "AUS" IST SPITZENREITER
--AUSG.------> DER TEXT KLINGT DANN ETWAS CHINESISCH.
--AUSG.------> DIE ANDEREN SACHEN HABEN WIR FAST ALLE ABGEFANGEN
--AUSG.------> DURCH ZWANGSWEISES HINFUEHREN

BENUTZER ---> Habt Ihr alle Nummern, die in eurer nummernliste stehen
--EING.------> selbst ausprobiert?
--EING.------> Duerfen wir die DECATES-Nummer im Artikel veroeffentlichen?
--EING.------>

CONSOL ------> AUSPROBIERT HABEN WIR KEINE (WIR WERDEN NUR ANGERUFEN)
--AUSG.------> VEROEFFENTLICHEN DUERFT IHR ALLE

BENUTZER ---> Habt ihr keine Moeglichkeiten anzurufen, keine Lust,
--EING.------> oder kein Budget?
--EING.------>

CONSOL ------> 1. WENN WIR ANRUFEN WUERDEN WAERE FUER DIESE ZEIT
--AUSG.------> DER ANSCHLUSS DER MAILBOX BLOCKIERT.
--AUSG.------> 2. IST ES NICHT ERWUENSCHT ABFRAGEN ZU MACHEN
--AUSG.------> 3. HABEN WIR KEINE ZEIT
--AUSG.------> 4. WENN MAN SO VIEL MIT DEM ZEUG ZU TUEN HAT IST JEDER
--AUSG.------> FROH MAL WAS ANDERES ZU MACHEN ALS IN BOXEN ZU SCHAUEN

BENUTZER ---> Mit wievielen unterhaltet ihr diese Mailbox?
--EING.------> Was sind eure Arbeitszeiten?
--EING.------> Wieviel d Freizeit steckt ihr rein?
--EING.------> Habt ihr schichtbetrieb/?
--EING.------> Ich schaetze, Das Du im Moment nicht bezahlt wirst, oder?
--EING.------>

CONSOL ------> 1. ICH UNTERHALTE DIE BOX ALLEINE
--AUSG.------> ARBEITSZEIT VON 9 BIS 16H
--AUSG.------> FREIZEIT CA 4 STD PRO TAG
--AUSG.------> SCHICHT NEIN
--AUSG.------> NICHT BEZAHLT IST RICHTIG
--AUSG.------> ICH HABE IM JANUAR EIN PROGRAMM BEGONNEN FUER DIESE ANWENDUNG
--AUSG.------> HABE DANN UNSER SYSTEM FUER DIESE ANWENDUNG ZUR VERFUEGUNG
--AUSG.------> GESTELLT BEKOMMEN UND BETREIBE DIES IN MEINER FREIZEIT
--AUSG.------> DA ICH NUR 2 TREPPEN BIS INS BUERO HABE.
```

Interessante Nummern von deutschen und ausländischen Mailboxen

Interview mit Decates

den Akustikkoppler — die Betriebslampe leuchtet. Alles ok. Leider meldet das Telefon das Besetztzeichen. Damit muß man immer rechnen. Also noch einmal. Diesmal klappt es. Ein hoher Pfeifton sagt mir, daß die Verbindung hergestellt ist. Schnell lege ich den Hörer auf den Akustikkoppler. Gespannt warte ich auf die ersten Zeichen. Nach ein paar Sekunden tut sich etwas. Die ersten Zeichen laufen mit der für 300 Baud typischen, langsamen Geschwindigkeit über den Bildschirm. Ein erstes Bild teilt mir mit, daß ich mit DECATES verbunden bin. Gleichzeitig werde ich nach meinem Paßwort gefragt. Ich gebe es ein und sofort kommt die Bestätigung, daß ich eingetragener Benutzer bin.

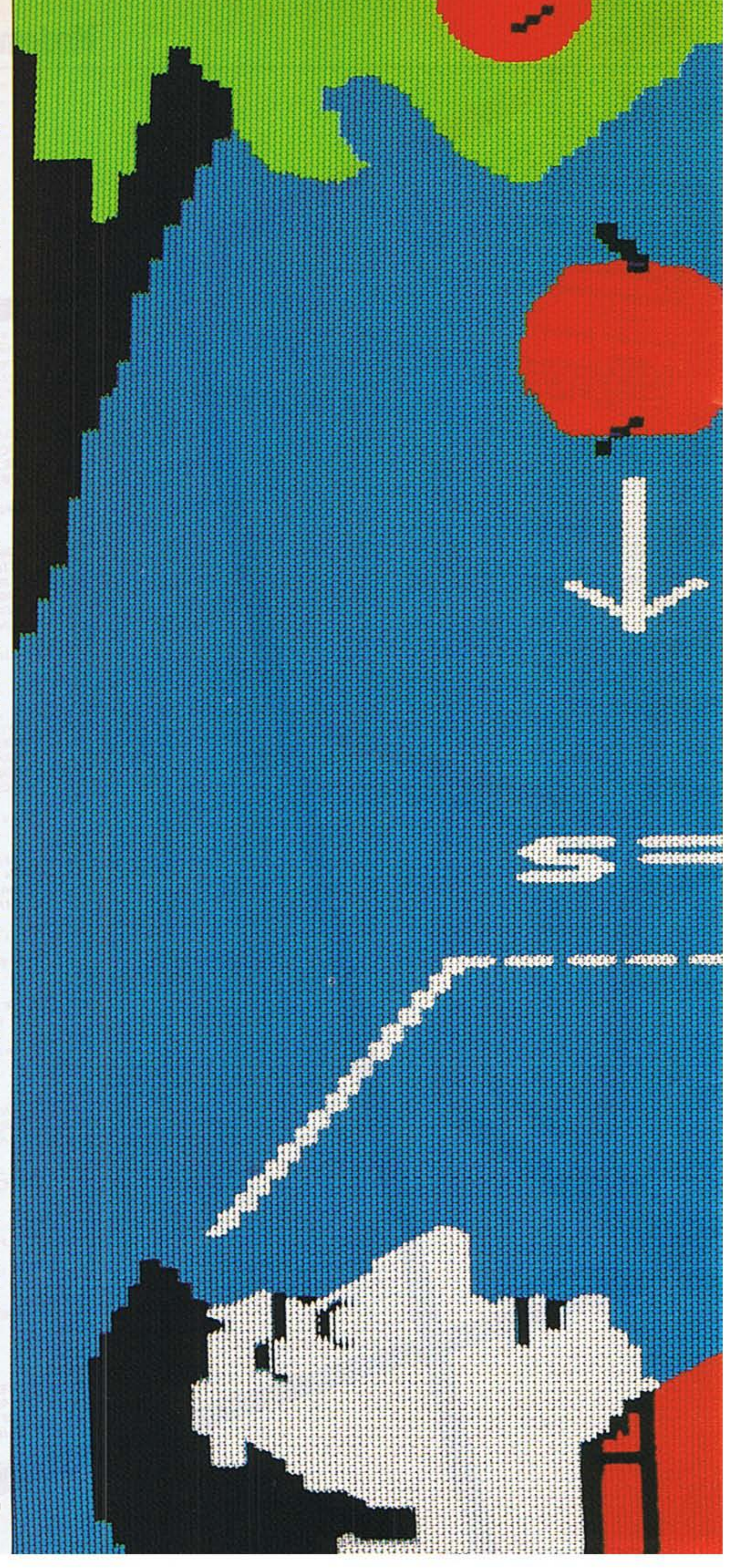
Als Anfänger werden Sie noch kein Paßwort besitzen. Aber keine Angst. Wer noch kein Paßwort besitzt, braucht nur RETURN zu drücken.

Als eingetragener Benutzer bin ich in der Lage, persönliche Mitteilungen abzurufen, die nur für mich bestimmt sind. Kein anderer kann diese Infos lesen. Jeder andere Benutzer kann jedoch jedem anderen Mitteilungen zukommen lassen und auch jede öffentliche Information entgegennehmen.

Die dann folgende Frage von DECATES nach der Bedienungsanleitung sollten Sie ruhig mit Ja beantworten, solange Sie sich mit diesem System nicht auskennen. Laut DECATES gibt es etliche Anrufer, die am Ende nicht mehr wissen, wie sie wieder aus dem System herauskommen (nämlich indem man LOGOFF eingibt). Die meisten Bedienungsfehler werden jedoch mittlerweile abgefangen. Der Rest geht automatisch. Und wie gesagt, die Bedienungsanleitung sollten Sie am Anfang immer neben sich liegen haben. Sie können dadurch Telefonkosten sparen.

Viele Mailboxen lassen folgende Funktionen zu:
Informationen eingeben, Informationen anzeigen; eine Art Fundgrube für Hard- und Softwaretausch ist oft auch vorhanden. Bei einigen Mailboxen existiert zusätzlich noch ein Informationssystem für Gewerbetreibende. Dort haben »gewöhnliche« Anrufer keinen Zugriff.
Probieren Sie die Mailboxen ruhig einmal aus. Weitere Nummern finden Sie in den abgedruckten Auszügen aus einer Verbindung mit DECATES. Doch vergessen Sie nicht die Telefonrechnung.

(gk)



WER ENTFÜHRT EUCH IN DIE WUNDERWELT DER WISSENSCHAFT?

$\frac{1}{2} \times g \times t^2$

COMMODORE COMPUTER.

Den einen führt der Commodore-Heimcomputer von den ersten Schritten der Physik in die grenzenlose Welt der Astrophysik. Den anderen von Bio und Chemie in die irdische Welt der Biochemie.

Ein faszinierendes Ding: ein echter Computer mit unbegrenzten Möglichkeiten. Mit ihm kann man spielend die Weltsprachen der Computer lernen. Kann man Daten, Adressen oder Plattensammlungen organisieren. Sogar videospielen kann man damit.

Ein tolles Ding: ein echter Computer für eine gute Idee nach der anderen. Der Commodore-Heimcomputer. Er kostet nicht die Welt.

Beim Commodore-Vertragshandel, in führenden Warenhäusern, guten Rundfunk- und Fernsehfachgeschäften und großen Versandhäusern.

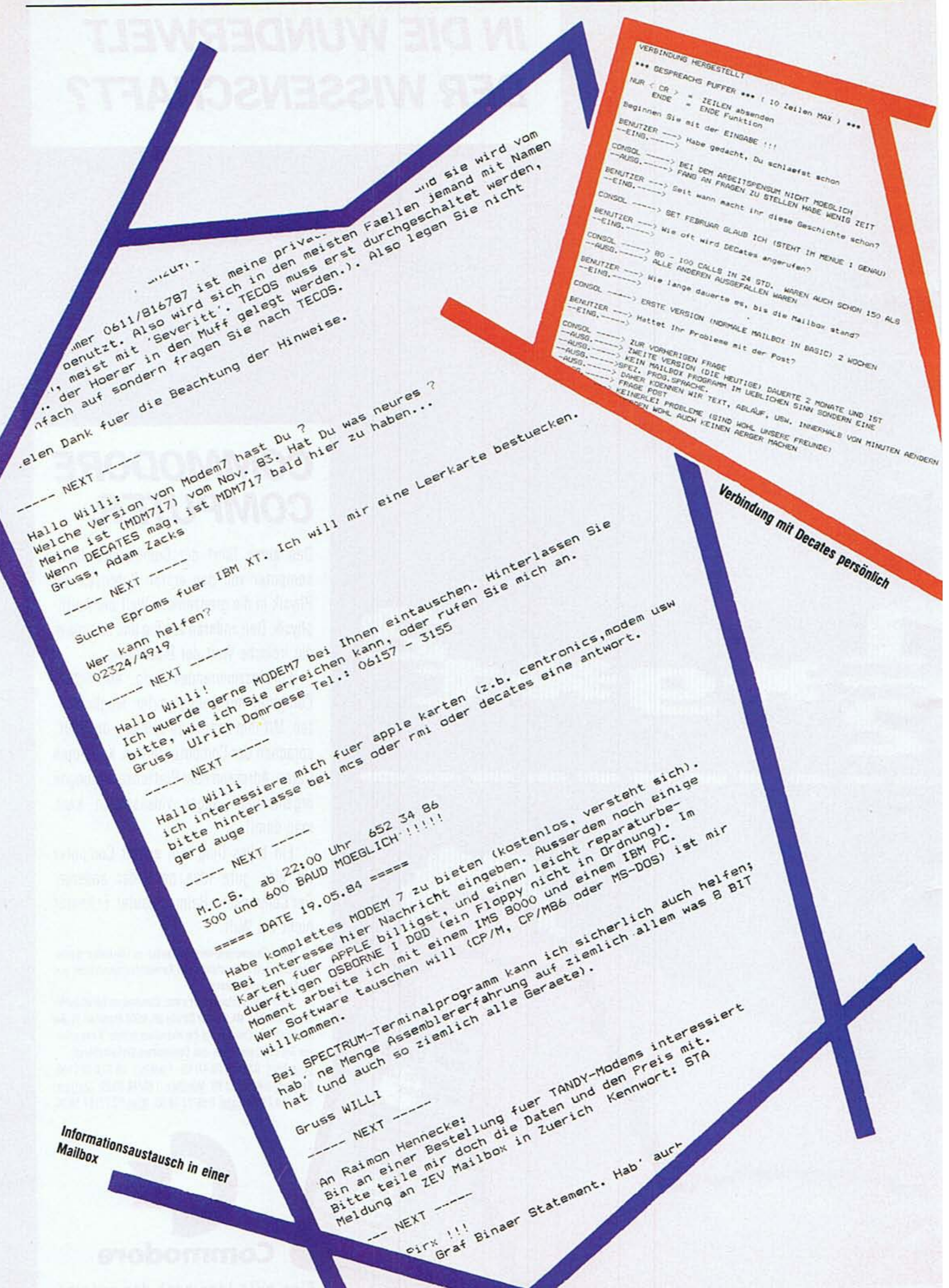
Mehr Informationen gibt's von: Commodore Büromaschinen GmbH, Abt. MK, Lyoner Straße 38, 6000 Frankfurt 71. Die Anschrift des Commodore-Fachhändlers in Ihrer Nähe erfahren Sie telefonisch von den Commodore-Verkaufsbüros: Düsseldorf 02 11/31 20 47/48, Frankfurt 06 11/6 63 81 99, Hamburg 0 40/21 13 86, München 0 89/46 30 09, Stuttgart 07 11/24 73 29, Basel 0 61/23 78 00, Wien 02 22/67 56 00.

COMMODORE AUF VIDEO:
„Einblick für Leute mit Weitblick"

Über 1 Stunde spannende Informationen, wie ein Computer funktioniert und was man alles damit machen kann. Video Cassette (180er Scotch Band) per Nachnahme oder per Scheck anfordern bei: Commodore GmbH - Video -, Postfach 260, 6082 Walldorf (Achtung: bitte Video-System angeben!) Einmalige Schutzgebühr incl. Versand nur DM 25,– zuzügl. Nachn.-Geb.



Eine gute Idee nach der anderen.



...mer 0611/816787 ist meine privat... ...nd sie wird vom
...benutzt. Also wird sich in den meisten Faellen jemand mit Namen
...', meist mit 'Severitt'. TECOS muss erst durchgeschaltet werden,
... der Hoerer in den Muff gelegt werden.). Also legen Sie nicht
...nfach auf sondern fragen Sie nach TECOS.

...elen Dank fuer die Beachtung der Hinweise.

----- NEXT -----

Hallo Willi:
Welche Version von Modem7 hast Du ?
Meine ist (MDM717) vom Nov. 83. Hat Du was neures ?
Wenn DECATES mag, ist MDM717 bald hier zu haben...
Gruss, Adam Zacks

----- NEXT -----

Suche Eproms fuer IBM XT. Ich will mir eine Leerkarte bestuecken.

Wer kann helfen?
02324/4919

----- NEXT -----

Hallo Willi!
Ich wuerde gerne MODEM7 bei Ihnen eintauschen. Hinterlassen Sie
bitte, wie ich Sie erreichen kann, oder rufen Sie mich an.
Gruss Ulrich Domroese Tel.: 06157 - 3155

----- NEXT -----

Hallo Willi
ich interessiere mich fuer apple karten (z.b. centronics,modem usw
bitte hinterlasse bei mcs oder rmi oder decates eine antwort.
gerd auge

----- NEXT -----

M.C.S. ab 22:00 Uhr 652 34 86
300 und 600 BAUD MOEGLICH !!!!!

===== DATE 19.05.84 =====

Habe komplettes MODEM 7 zu bieten (kostenlos, versteht sich).
Bei Interesse hier Nachricht eingeben. Ausserdem noch einige
Karten fuer APPLE billigst, und einen leicht reparaturbe-
duerftigen OSBORNE I DOD (ein Floppy nicht in Ordnung). Im
Moment arbeite ich mit einem IMS 8000 und einem IBM PC.
Wer Software tauschen will (CP/M, CP/M86 oder MS-DOS) ist mir
willkommen.

Bei SPECTRUM-Terminalprogramm kann ich sicherlich auch helfen;
hab' ne Menge Assemblererfahrung auf ziemlich allem was 8 BIT
hat (und auch so ziemlich alle Geraete).

Gruss WILLI

----- NEXT -----

An Raimon Hennecke:
Bin an einer Bestellung fuer TANDY-Modems interessiert
Bitte teile mir doch die Daten und den Preis mit.
Meldung an ZEV Mailbox in Zuerich Kennwort: STA

-- NEXT -----

Pirx !!!
Graf Binaer Statement. Hab' auch

Informationsaustausch in einer Mailbox

VERBINDUNG HERGESTELLT
*** GESPREACHS PUFFER *** (10 Zeilen MAX) ***
NUR < CR > = ZEILEN absenden
ENDE = ENDE Funktion
Beginnen Sie mit der EINGABE !!!
BENUTZER ---> Habe gedacht, Du schlaefst schon
--EING.----->
CONSOL -----> BEI DEM ARBEITSPENSUM NICHT MOEGLICH
--AUSG.-----> FANG AN FRAGEN ZU STELLEN HABE WENIG ZEIT
BENUTZER ---> Seit wann macht ihr diese Geschichte schon?
--EING.----->
CONSOL -----> SET FEBRUAR GLAUB ICH (STEHT IM MENUE 1 GENAU)
BENUTZER ---> Wie oft wird DECates angerufen?
--EING.----->
CONSOL -----> 80 - 100 CALLS IN 24 STD. WAREN AUCH SCHON 150 ALS
--AUSG.-----> ALLE ANDEREN AUSGEFALLEN WAREN
BENUTZER ---> Wie lange dauerte es, bis die Mailbox stand?
--EING.----->
CONSOL -----> ERSTE VERSION (NORMALE MAILBOX IN BASIC) 2 WOCHEN
BENUTZER ---> Hattet Ihr Probleme mit der Post?
--EING.----->
CONSOL -----> ZUR VORHERIGEN FRAGE
--AUSG.-----> ZWEITE VERSION (DIE HEUTIGE) DAUERTE 2 MONATE UND IST
--AUSG.-----> KEIN MAILBOX PROGRAMM IM UEBLICHEN SINN SONDERN EINE
--AUSG.----->SPEZ. PROG.SPRACHE.
--AUSG.-----> DAHER KOENNEN WIR TEXT, ABLAUF, USW. INNERHALB VON MINUTEN AENDERN
--AUSG.-----> FRAGE POST
... KEINERLEI PROBLEME (SIND WOHL UNSERE FREUNDE)
... WERDEN WOHL AUCH KEINEN AERGER MACHEN

Verbindung mit Decates persönlich

FORTH – die etwas andere Programmiersprache

Stellen Sie sich eine Programmiersprache vor, interaktiv wie Basic, schnell wie Assembler, strukturiert wie Pascal und beliebig erweiterbar. Eine solche Sprache ist Forth.

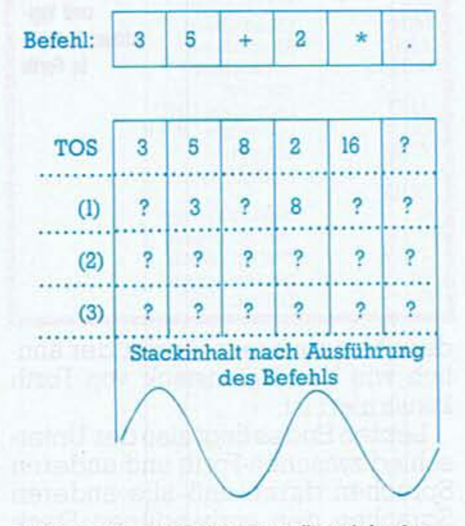
Forth wurde Anfang der siebziger Jahre von Charles H. Moore entwickelt und ursprünglich zur Steuerung von Radioteleskopen eingesetzt. Die Sprache entstand dabei ganz gezielt als Alternative zur fehleranfälligen und schlecht zu wartenden Assemblerprogrammierung auf der einen Seite und den klassischen Compilersprachen wie Fortran oder Algol auf der anderen Seite, die für Prozeßsteuerungen zu langsam und zu aufwendig waren.

Forth wurde übrigens damals als sogenannte »Programmiersprache der vierten Generation« entwickelt und sollte eigentlich dementsprechend den Namen »Fourth« tragen. Leider war die IBM 1130, auf der Forth zum ersten Mal implementiert wurde, noch ein Rechner der »dritten Generation« und erlaubte nur Dateinamen bis maximal fünf Zeichen. Charles Moore mußte »Fourth« daher um einen Buchstaben kürzen, und das war die Geburtsstunde von Forth. Lange Jahre fristete die Sprache ein Schattendasein in diversen Forschungsstätten, ehe sie mit dem Aufkommen der Mikrocomputer eine größere Verbreitung auch unter Privatleuten fand.

Was ist Forth?

Forth ist interaktiv. Ähnlich wie in Basic können Programme also im Dialog mit dem Computer entwickelt und getestet werden. Das ist ein wesentlicher Unterschied zu den klassischen Compilersprachen wie zum Beispiel Pascal oder Fortran. Ein Compiler erwartet nämlich ein fertiges Programm, das er dann in einem Arbeitsgang übersetzt. Es gibt keine Möglichkeit, die Wirkung bestimmter Kommandos vor der Übersetzung zu testen. Anders bei Forth. Hier kann man im Direktmodus Rechnungen durchführen, Programmteile testen und neue Befehlsworte definieren.

Forth ist strukturiert. Es gibt in



TOS	3	5	8	2	16	?
(1)	?	3	?	8	?	?
(2)	?	?	?	?	?	?
(3)	?	?	?	?	?	?

Bild 1. Die Stapelveränderungen während der Ausführung einer einfachen Befehlsfolge. Die Operationen »+« und »*« verknüpfen jeweils die beiden obersten Elemente des Stacks miteinander und legen das Ergebnis wieder ab. Mit ».« wird der TOS ausgedruckt und steht danach nicht mehr zur Verfügung.

Forth selbst mit Gewalt keine Möglichkeit, Basic-ähnlichen Spaghetti-Code zu erzeugen. Der Grund dafür ist sehr einfach: Es gibt keine GOTO-Befehle und auch keinen Ersatz dafür. Jede Wort-Definition ist in sich abgeschlossen und am ehesten noch mit den Prozeduren in Pascal zu vergleichen.

Forth ist schnell. Alle Anweisungen werden zuerst compiliert und dann ausgeführt, was insbesondere bei Programmschleifen große Zeitvorteile bringt. Forth-Programme sind daher in der Regel um Größenordnungen schneller als entsprechende Basic-Programme.

Forth ist erweiterbar. Im Gegensatz zu den meisten anderen Programmiersprachen kann der Benutzer in Forth neue Sprachelemente definieren, ja Programmierung in Forth besteht gerade in der Definition neuer Worte zur Erweiterung des Sprachumfangs.

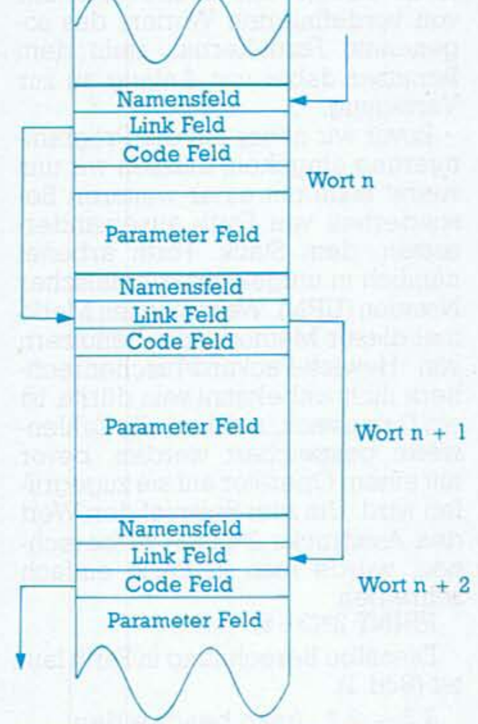


Bild 2. Vereinfachte Darstellung des Forth-Wörterbuchs im Speicher. Das Namensfeld enthält den Wortnamen, das Codefeld beinhaltet Information über die Art des folgenden Parameterfeldes, das die Definition des Wortes enthält. Ein Wort kann als Maschinenspracheabschnitt, als Variable oder Konstante oder — der häufigste Fall — durch andere Befehlsworte definiert sein (Colon-Definition). Das Link-Feld schließlich verkettet die Wortdefinitionen in Forth zu einer Liste, die bei der Interpretation oder Compilation von Worten durchsucht wird.

Jeder Basic-Programmierer wird erfreut zur Kenntnis nehmen, daß in Forth keine Fehlermeldung »SYNTAX ERROR« existiert — und zwar ganz einfach deswegen, weil die Forth-Syntax so einfach ist, daß keine solchen Fehler vorkommen können. Ein Wort in Forth besteht aus einer beliebigen Zeichenfolge, die allerdings kein Leerzeichen enthalten darf, weil dieses gerade zwei Worte trennt. Zum Beispiel sind gültige Forth-Worte

HALLO, 3FACH, %X!, +1NOCH-UND-DANN-SCHLUSS.

Eine Eingabezeile besteht nun einfach aus Worten und Zahlen. Bei Zahlen ist übrigens noch eine Besonderheit zu beachten: Forth kann

in beliebigen Zahlensystemen rechnen. Die Systemvariable BASE enthält die jeweils gültige Zahlenbasis. In der Regel wird man entweder im Dezimal- oder im Hexadezimalsystem arbeiten. Daher kennt Forth zwei spezielle Befehlsworte zum umschalten in das jeweilige Zahlensystem, nämlich DECIMAL und HEX.

Programmierung in Forth besteht nun einfach darin, mittels bereits definierter Worte wiederum neue Worte zu bilden. Ein Grundvokabular von vordefinierten Worten, das sogenannte Forth-Kernal, steht dem Benutzer dabei von Anfang an zur Verfügung.

Bevor wir näher auf die Programmierung eingehen, müssen wir uns zuerst noch mit einer weiteren Besonderheit von Forth auseinandersetzen, dem Stack. Forth arbeitet nämlich in umgekehrter polnischer Notation (UPN). Wesentliches Merkmal dieser Methode, die Benutzern von Hewlett-Packard-Taschenrechnern nicht unbekannt sein dürfte, ist ein Datenstack, auf dem alle Zahlenwerte gespeichert werden, bevor mit einem Operator auf sie zugegriffen wird. Um zum Beispiel den Wert des Ausdrucks 2*(3+5) zu berechnen, würde man in Basic einfach schreiben

PRINT 2*(3+5)

Dieselbe Berechnung in Forth lautet (Bild 1):

3 5 + 2 * . (man beachte den Punkt !)

Die Berechnung geht also genauso vor sich, wie man auch im Kopf rechnen würde: Zunächst nimmt man die beiden Zahlen in der Klammer, weil diese zuerst berechnet werden. Dann werden die Zahlen addiert, schließlich holt man sich die »2« und multipliziert sie mit dem ersten Zwischenergebnis. Forth legt also generell erst die für einen Befehl benötigten Operanden auf den Stack, und der Befehl greift anschließend darauf zu. Das gilt nicht nur für arithmetische Operationen, sondern ist ein Kennzeichen für die gesamte Sprachstruktur von Forth. Nehmen wir etwa den Punkt am Ende der Beispielrechnung oben. Er ist das Forth-Äquivalent zum PRINT-Befehl in Basic.

Beachten Sie bitte, daß der PRINT-Befehl immer vor den auszudruckenden Daten steht, der Punkt in Forth jedoch immer dahinter. Mit dem Punkt-Befehl wird stets der oberste Zahlenwert auf dem Stack ausgedruckt. Der wesentliche Unterschied zu anderen Programmiersprachen, wie zum Beispiel Basic, liegt nun darin, daß der Operand zum Zeitpunkt des Erreichens des Punkt-Befehls bereits bekannt ist. Der Basic-Interpreter, der auf einen PRINT-Befehl stößt, weiß zu diesem Zeitpunkt ja noch gar nicht, was er ausdrucken soll. Also muß die Information, daß gedruckt werden soll, irgendwo abgespeichert werden. Basic und alle anderen höheren Programmiersprachen bedienen sich dazu eines internen Stacks, der ähnlich wie der Datenstack von Forth konstruiert ist.

+	Addition
–	Subtraktion
*	Multiplikation
/	Division
MOD	Divisionsrest
/MOD	Rest und Ergebnis bei Division
MAX	Maximalwert
MIN	Minimalwert
ABS	Absolutwert
MINUS	Vorzeichenwechsel
AND	Logisches UND
OR	Logisches ODER
XOR	Logisches EXCLUSIV
ODER	
<	Test ob kleiner
>	Test ob größer
=	Test ob gleich
0<	Test ob negativ
0=	Test ob Null

Tabelle 1. Die wichtigsten arithmetischen und logischen Befehle in Forth

Letzten Endes liegt also der Unterschied zwischen Forth und anderen Sprachen darin, daß alle anderen Sprachen den vorhandenen Stack vor dem Benutzer verheimlichen und zu diesem Zweck natürlich zusätzlichen Verwaltungsaufwand treiben müssen, was die Verarbeitung nicht gerade beschleunigt.

Da Forth mit einem Datenstack arbeitet, funktionieren alle eingebauten und selbstdefinierten Funktionen nach dem gleichen Schema: Die Funktion holt sich die benötigten Parameter vom Stack, führt die notwendigen Berechnungen durch und legt das Ergebnis wieder auf den Stack. Dadurch können Funktionsaufrufe beliebig geschachtelt werden. Eine Übersicht über die wichtigsten numerischen Funktionen in Forth gibt Tabelle 1.

Da der Stack bereits für die laufenden Rechnungen benötigt wird, ist es einigermaßen umständlich, ihn auch noch für die Speicherung von Daten zu verwenden. Forth kennt daher die beiden Befehlsworte VARIABLE und CONSTANT, die zur Definition von Variablen beziehungsweise Konstanten benutzt werden. Mit 4 CONSTANT VIER wird zum Beispiel eine Konstante mit Namen VIER und Wert 4 angelegt, auf die im weiteren Verlauf immer wieder zurückgegriffen werden kann. Wann immer Forth jetzt auf das Wort VIER stößt, wird die Zahl 4 auf den Stack gelegt.

Bei Variablen ist die Situation etwas komplizierter. Sei eine Variable TEST mit Anfangswert 1 definiert worden durch 1 VARIABLE TEST. Wenn jetzt wieder das Wort TEST auftaucht, wird nicht der Inhalt der Variablen, sondern deren Adresse auf den Stack gelegt. Mit zwei sehr häufig gebrauchten Befehlen kann man nun auf diese Adresse zugreifen: Es sind dies die Befehle »!« (gesprochen »Store«) und »@« (gesprochen »Fetch«). Der Befehl »!« speichert den Zahlenwert, der an zweiter Stelle auf dem Stack liegt, an die Adresse, die durch das oberste Element des Stapels (TOS = Top of Stack) gegeben ist. Beide Zahlenwerte werden dabei vom Stack entfernt. Die Wirkung ist analog zum POKE-Befehl in Basic, operiert aber mit 16-Bit-Worten statt mit einzelnen Bytes. 2 840 ! entspricht also POKE 840,2:POKE 841,0 in Basic. Ebenso ist 840 @ in Forth analog zur Basic-Funktion PEEK(840)+256*PEEK (841). Sollen tatsächlich nur einzelne Bytes gespeichert und gelesen werden, bedient man sich der Worte »C!« und »@«. 1 840 C! entspricht POKE 840,1 in Basic und 840 C @ entspricht PEEK(840).

Nun sollte auch das Arbeiten mit Forth-Variablen klar sein: Bei jedem Auftreten eines Variablennamens legt Forth die Adresse dieser Variablen auf den Stack. Um zum Beispiel einer Variablen TEST den Wert 7 zuzuweisen, schreibt man in Forth

7 TEST !

Neue Worte definieren

Danach ist die Zahl 7 und auch die Adresse von TEST vom Stack verschwunden. Benötigt man die Zahl jedoch noch für weitere Rechnungen, muß sie vor der Zuweisung an die Variable zunächst dupliziert werden. Hierzu dient das Wort DUP, welches ein Duplikat des TOS erzeugt. DUP ist eine sehr häufig benutzte Funktion, da alle Forth-Befehle die Parameter vom Stack entfernen. Will man zum Beispiel das Zwischenergebnis einer längeren Rechnung mit dem Befehl ».« ausdrucken, dann muß es zunächst mit DUP dupliziert werden, da es durch ».« vom Stack gelöscht wird und somit für weitere Rechnungen nicht mehr zur Verfügung steht.

Forth hat die bemerkenswerte Eigenschaft, daß der Sprachumfang

beliebig erweitert werden kann. Zwei Möglichkeiten zur Schaffung neuer Worte haben wir ja schon kennengelernt: CONSTANT und VARIABLE. Daneben gibt es noch eine Reihe weiterer sogenannter Definitionsworte. Zum Beispiel können mit CREATE neue Forth-Worte direkt als Maschinenspracheroutinen definiert werden (sogenannte »Primitives«). Man wird davon allerdings nur an besonders zeitkritischen Stellen Gebrauch machen und in der Regel neue Worte mittels der sogenannten »Colon-Definition« einführen. Die Syntax dieser Definitionsmethode ist die folgende:

: NEUWORT ALTWORT1 ALTWORT2 .... ALTWORTn ;

Diese Befehlssequenz erzeugt ein neues Wort, NEUWORT, welches durch die bereits vorhandenen Worte ALTWORT1 bis ALTWORTn definiert ist. Die Colon-Definition beginnt also immer mit dem Wort »:« und endet mit dem Wort »;«. Es ist wichtig, sich genau klar zu machen, daß »:« und »;« nicht irgendwelche syntaktisch notwendigen Zeichen sind, sondern daß es sich dabei tatsächlich um normale Forth-Worte handelt. Das Wort »:« ruft vereinfacht gesagt den Forth-Compiler auf, während das Wort »;« eine Anweisung zum Beenden der Compilation darstellt.

Hinsichtlich der Namensgebung für neue Worte ist der Anwender im Prinzip keinen Beschränkungen unterlegen. Benötigt man zum Beispiel des öfteren den Durchschnitt zweier Zahlen, dann kann man einfach definieren

: MITTEL + 2 / . ;

Nach dieser Definition kann man jetzt einfach schreiben 4 8 MITTEL und erhält den Wert 6 angezeigt. MITTEL erwartet also zwei Zahlen auf dem Stack, die zunächst addiert werden (»+«). Das Ergebnis dieser Operation wird dann durch zwei geteilt (»2 /«) und schließlich ausgedruckt (».«). An der Definition von MITTEL ist schon zu sehen, wie kurz und effizient Forth-Programme sein können. In Basic würde ein entsprechendes Programm etwa folgendermaßen aussehen:

10 INPUT X,Y : PRINT (X+Y)/2 : REM Mittel

Mit der Colon-Definition kann man auch auf einfachem Wege Forth-Worten neue Namen zuweisen. Wenn man zum Beispiel für das Ausdrucken einer Zahl lieber den Befehl »DRUCKE« hätte, kann man einfach definieren

: DRUCKE . ;

Nun kann man überall DRUCKE statt ».« schreiben. Wenn man es wieder leid ist, ständig sechs Buchstaben statt einem schreiben zu müssen, teilt man dem Forth-System einfach mit, daß es das Wort DRUCKE vergessen soll. Der dazu nötige Befehl lautet FORGET DRUCKE. Allerdings löscht dieser Befehl nicht nur die Definition von DRUCKE, sondern auch alle anderen eventuell noch danach erfolgten Definitionen. Das hängt damit zusammen, daß Forth alle Definitionen in einer verketteten Liste abspeichert, dem sogenannten Wörterbuch (Bild 2).

Strukturiert Programmieren

Forth zwingt den Programmierer geradezu, seine Programme modular aufzubauen. Es gibt in Forth keinen dem GOTO in Basic vergleichbaren Befehl. Ein komplexes Forth-Programm entsteht immer zuerst auf dem Papier. Wenn die Programmstruktur festliegt, entwickelt man zunächst die benötigten Unterroutinen und testet sie aus. Diese Unterroutinen sind danach ja als völlig normale Forth-Worte zu benutzen und können dazu verwendet werden, nun eine weitere Ebene noch »mächtigerer« Worte zu definieren. Am Ende dieses Prozesses steht dann praktisch ein Wort, das das gesamte Programm repräsentiert.

Forth besitzt, ähnlich wie Pascal, eine ganze Reihe sogenannter Kontrollstrukturen, um strukturierte Programmierung zu unterstützen:

IF ... ENDIF — Die IF-Abfrage testet, ob der TOS (Top of Stack, also das oberste Element des Stacks) den Wert Null (= False) oder einen Wert ungleich Null (= True) hat. Ist das Ergebnis »True«, dann werden die Anweisungen zwischen IF und ENDIF ausgeführt, andernfalls wird das Programm nach ENDIF fortgesetzt. Einige ältere Forth-Versionen benutzen das Schlüsselwort THEN statt ENDIF.

IF ... ELSE ... ENDIF — Arbeitet ähnlich der einfachen IF-Abfrage, führt aber im Falle TOS=0 die Anweisungen zwischen ELSE und ENDIF aus, sonst die Anweisungen zwischen IF und ELSE.

DO ... LOOP — Entspricht der FOR-NEXT-Anweisung in Basic. Das Wort DO erwartet zwei Parameter auf dem Stack, nämlich die Anfangs- und die Endzahl der Schleifendurchläufe. Mit LOOP wird die Schleife beendet. Falls das Wort +LOOP statt LOOP verwendet wird, erhöht sich der Schleifenindex nicht um 1, sondern um den Wert des TOS. Das Wort »I« wird innerhalb von Schleifen dazu verwendet, den Schleifenzähler auf den Stack zu kopieren.

BEGIN ... UNTIL — Diese Schleife wird solange durchlaufen, bis UNTIL einen Wert ungleich Null (also »True«) auf dem Stack vorfindet. Entspricht REPEAT ... UNTIL in Pascal.

BEGIN ... WHILE ... REPEAT — Der Programmteil zwischen BEGIN und REPEAT wird solange durchlaufen, bis das Wort WHILE die Bedingung TOS=0 feststellt. Danach wird die Schleife unmittelbar hinter WHILE verlassen.

BEGIN ... AGAIN — Dient zum Programmieren unendlicher Schleifen und testet keinerlei Bedingungen.

Wer in seinen Forth-Programmen weitere Schleifen- oder Auswahlstrukturen benötigt, kann sie sich einfach selber definieren. Das gilt für praktisch alle Bereiche. Forth arbeitet zum Beispiel normalerweise nur mit Integer-Arithmetik. Wer nun für seine spezielle Anwendung Real-Zahlen benötigt, besorgt sich einfach ein REAL-Vokabular von Forth und bindet es in sein System ein. So kann sich jeder Anwender sein persönliches Forth-System zusammenstellen und nach seinen Bedürfnissen von Fall zu Fall erweitern.

Es ist klar, daß es sehr schwierig ist, für eine derartig flexible Sprache einen sinnvollen Standard zu finden. Die Forth Interest Group (FIG), eine nichtkommerzielle Vereinigung von Forth-Enthusiasten, bemüht sich seit Jahren mit gewissem Erfolg um eine Standardisierung von Forth. Zu diesem Zweck gibt die Forth Interest Group vollständige Assemblerlistings des sogenannten FIG-Forth mit Implementationshinweisen für alle gängigen Mikrocomputer heraus. Eine spezielle Forth-Zeitschrift, die »Forth Dimensions« kann dort ebenfalls bezogen werden. Die Adresse ist Forth Interest Group, P.O. Box 1105, San Carlos, CA 94070, USA.

Wer sich näher mit Forth beschäftigen will, dem seien zum Abschluß noch drei Bücher empfohlen:

Starting Forth von Leo Brodie, zu beziehen über die Forth Interest Group;

Das Forth Handbuch von H. Floegel, erschienen im Hofacker-Verlag;

Die Programmiersprache Forth von R. Zech, erschienen im Franzis-Verlag.

(ev)

CP/M-Software auf den Comm…



Konnten Sie bisher mit Ihrem CP/M-Modul wenig anfangen? Versuchten Sie vergeblich Programme dafür zu kaufen? Dieser Bericht zeigt Ihnen einen Weg zur Welt der CP/M-Software.

Stellen Sie sich vor, Sie schalten den Commodore 64 an, laden ein Programm und Ihr Computer meldet sich mit »WORDSTAR release 3.25«. Der C 64 als Computer für Programme, die für wesentliche größere (und teurere) Systeme entwickelt wurden! Oder möchten Sie in Fortran, Cobol, PL1 oder Algol programmieren? Warum nicht? Das zur Verfügung stehende Potential an CP/M-Software ist beinahe unerschöpflich.

Aber woher ist diese Software zu beziehen? Diese Frage ist berechtigt. Wer einmal versucht hat, in einem Fachgeschäft, oder einem Softwarehaus CP/M-Programme im Diskettenformat der 1541-Floppys zu kaufen, wurde wahrscheinlich enttäuscht.

Eigentlich schade, denn eine Umfrage bei einigen Münchener Computerfachgeschäften erbrachte, daß nach dem Steckmodul häufig gefragt wurde. Erhältlich war das Modul aber nur in einem einzigen Geschäft. Dort wurde es, der eigenen Stellung wohl bewußt, nur zögernd unter dem Ladentisch hervorgezogen. Erst nach mehrmaligem Betonen der Dringlichkeit des Kaufs, konnte sich der Verkäufer zum Tausch des Moduls gegen Geld entschließen. Ein Markt, der einfach brachliegt, beziehungsweise einfallsreichen Bastlern überlassen wird (der Diplomkaufmann in mir regt sich).

Nutzen wir die Zeit, bis es eigene C 64-CP/M-Software gibt, mit einer gar nicht so schlechten Behelfslösung. Wir koppeln einen Apple II mit dem Commodore. Dabei geht man wie folgt vor:

Punkt 1: Sie bauen oder kaufen sich ein Verbindungskabel zwischen einem Apple II und dem Commodore 64.

Punkt 2: Sie kaufen sich die gewünschte CP/M-Software im Apple II Format.

Punkt 3: Sie suchen einen Apple-II-Besitzer, (zum Beispiel über eine Anzeige).

Punkt 4: Sie schreiben sich, oder was leichter ist, sie kaufen sich Software zur Übertragung.

Punkt 5: Sie koppeln die beiden Computer, und überspielen die gekauften Programme.

Dieses Verfahren ist so einfach, wie wirkungsvoll.

Zum Test verwendete ich das Transferpaket von Bieling und machte mich an die Arbeit. Das Kabel wird auf der Apple-Seite in den Spieleanschluß gesteckt und beim Commodore natürlich in den User-Port. Nach dem Laden der Transfersoftware kann das Übertragen beginnen. »Schon« nach 18 Minuten war eine gesamte Diskette übertra-



zu erzielen. Dazu ist es notwendig, dieses näher zu betrachten. In dem Bereich $0B97 bis $0BCB befindet sich eine Routine, die zum Initialisieren des Laufwerkes, sowie zum Eröffnen einer Direktzugriffsdatei dient. Ferner wird in den Speicherstellen $0B9E und $0BB8 die Geräteadresse in das X-Register geladen. Damit liegt es nahe, diese Routine jedesmal, wenn die Laufwerksnummer (0 oder 1) gewechselt wird, anzuspringen und vorher das Argument LDX der Laufwerksnummer entsprechend auf 8 oder 9 zu setzen. Die Stelle, die die Laufwerksnummer überprüft, liegt im BIOS bei $0AF5. Dort wird die Diskettennummer ausgelesen, in ASCII-Code umgewandelt und gespeichert. Hier ersetzt man den Sprung in die ASCII-Wandelroutine durch den in eine Erweiterungsroutine zwischen $0C98 bis $0CBB. Falls die Laufwerksnummer 0 war, wird dort überprüft, ob das Argument von LDX 8 ist. Wenn ja, erfolgt ein Rücksprung. Wenn nein, wird es in 8 geändert und die Initialisierungsroutine angesprungen. Das gleiche gilt für den umgekehrten Fall.

Am komfortabelsten läßt sich diese Änderung mit dem Basic-Pro-

gen. Mit dem DOS im Floppy 1541 lassen sich leider ohne größeren Aufwand keine höheren Transferraten erreichen. Aber immerhin, es geht.

Eines sei bei aller Freude über diese Quelle wirklich guter Software angefügt: Da Computer und Programme immer jenseits von Gut und Böse stehen und somit nicht fähig sind, Urheberrechten und ähnlichem Beachtung zu schenken, obliegt Ihnen diese Pflicht. Erkundigen Sie sich lieber vorher, ob es Einwände von seiten der Lizenzgeber gegen die Übertragung gibt.

Was passiert aber, wenn es Ihnen gelungen ist, ein Programm zu kopieren. Läuft es einwandfrei oder sind da und dort noch kleine Änderungen vorzunehmen? Das erste Hindernis, dem sie auf Ihrer Erkundungsreise durch die CP/M-Software begegnen werden, ist deren Verlangen nach einer Darstellung von 80 Zeichen auf dem Bildschirm.

Hier empfiehlt es sich, die Anschaffung einer 80-Zeichen-Karte in Erwägung zu ziehen. Alle mir bekannten Softwarelösungen liefern ein unbefriedigendes Ergebnis.

Ein wesentlich schwerwiegenderes Problem liegt im BIOS (Basic-Input/Output-System) des Commodore-CP/M. Dieses ist bekanntlich für nur ein Laufwerk konzipiert worden. Viele CP/M-Programme setzen ihrerseits aber ein Doppellaufwerk voraus. Eine Lösung ist nur durch eine Abänderung des BIOS

gramm vornehmen (Programm laden, CP/M-Diskette in das Laufwerk einlegen, Programm starten). Vor dem Booten des CP/M-Ladeprogramms ist natürlich dafür zu sorgen, daß eines der Laufwerke die Geräteadresse 9 hat. Außerdem muß nach dem Starten des CP/M mit dem Programm Config (ist auf der Systemdiskette) die Anzahl der Laufwerke auf zwei erhöht werden. Nach dem Verlassen dieses Programms können Sie mit Ihren beiden Laufwerken wie mit einem Doppellaufwerk unter CP/M arbeiten.

Beispielsweise mit Wordstar, einem sehr beliebten und oft verkauften Textverarbeitungsprogramm.

Da ich die meisten meiner Berichte auf einem IBM-PC mit Wordstar schreibe, war ich natürlich sehr gespannt, wie das gleiche Programm auf dem Commodore 64 aussieht. Die erste Einschränkung, die ich machen mußte, war der Anspruch auf schnelle Diskettenoperationen. Viele der komfortablen Befehle des Wordstar gehen durch die langsame Ladegeschwindigkeit der Floppy verloren oder verlieren ihren Sinn.

Auch mit zwei Laufwerken bleibt das Laden, Speichern und Booten eine langwierige, Geduld erfordernde Angelegenheit. Zusätzlich fiel mir auf, daß eine Ausgabe auf den Drucker nur mit einem am seriellen Bus angeschlossenen Gerät möglich war. Wer aber beispielsweise einen Epson-Drucker mit einer Softwareschnittstelle betreibt, wird in den meisten Fällen keinen Buchstaben auf das Papier bekommen. Beachtet man den Preis, den man für eine Wordstar-Diskette im Apple-Format bezahlen muß, addiert dazu die Kosten für eine 80-Zeichen-Karte und das Modul, so lohnt sich fast schon die Anschaffung eines anderen Systems.

Die schon aus dem Handbuch zum Commodore 64 bekannte spärliche Dokumentation findet im Beipackzettel zum CP/M-Modul eine Steigerung. Mit den Hinweisen kann jeder Anwender etwa soviel anfangen, wie ein Fernsehzuschauer mit dem Pogramm der letzten Woche — nämlich nichts. Dabei werden auf der dem Modul beiliegenden Systemdiskette einige recht interessante Hilfsprogramme mitgeliefert. Außer einem lapidaren Hinweis auf etwa zur Verfügung stehende (und extra zu bezahlende) Literatur wird der frischgebackene Modulbesitzer mit einem windigen Blättchen Papier alleingelassen.

Obwohl es nicht die Aufgabe einer Zeitschrift ist, fehlende Handbücher nachzuliefern, sollen diese Dienstprogramme hier kurz vorgestellt werden:

```
1 REM AENDERUNG DES BIOS
2 REM ZUM BETRIEB DES
3 REM CP/M MODULS MIT
4 REM ZWEI LAUFWERKEN
5 REM
10 OPEN1,8,15:OPEN2,8,2,"#"
20 PRINT#1,"U1 2 0 1 1"
30 PRINT#1,"B-P 2 249"
40 PRINT#2,CHR$(152);CHR$(12);
50 PRINT#1,"U2 2 0 1 1"
60 CLOSE1:CLOSE2
70 OPEN1,8,15:OPEN2,8,2,"#"
80 PRINT#1,"U1 2 0 1 3"
90 PRINT#1,"B-P 2 152"
100 FORI=1TO36
110 READQ
120 PRINT#2,CHR$(Q);
130 NEXTI
140 PRINT#1,"U2 2 0 1 3"
150 CLOSE1:CLOSE2
160 DATA 240,15,162,9,236,159,11,240
170 DATA 24,142,159,11,142,185,11,16
180 DATA 13,162,8,236,159,11,240,9
190 DATA 142,159,11,142,185,11,32,151
200 DATA 11,169,0,96
READY.
```

Basicprogramm zum Ändern des BIOS

1. MOVCPM Mit diesem Programm können Sie Ihr Betriebssystem ändern, zum Beispiel an einen anderen Platz im Speicher verlegen. Die Routine kann dazu verwendet werden, eine 48 KByte-Version zu erzeugen, und mit SYSGEN direkt auf die CP/M-Spuren zu schreiben. Leider funktioniert dieses Dienstprogramm nicht fehlerfrei, denn es tritt ein Synchronisationsfehler auf. Wahrscheinlich stimmen die Versionen von CP/M und MOVCPM nicht überein. Eine 48-KByte-Version sollte immer dann konstruiert werden, wenn der Drucker und die Floppy am seriellen Bus angeschlossen ist. Für das Arbeiten mit einem IEEE-Interface muß man aber eine 44-KByte-CP/M-Version benutzen.

2. DDT. Diese sehr wichtige Routine dient der Kontrolle neuer Programme, deren sicheres »Laufen« noch ungewiß ist. Im einzelnen lassen sich damit Dateien lesen, Speicherinhalte anschauen und verändern sowie das Programmverhalten untersuchen.

3.SUBMIT Ein Programm, das für oft wiederkehrende Aufgabenstellungen sehr nützlich sein kann. Die Formulierung der Aufgabe mit dem Editor (ED) ist ausreichend; die Abarbeitung der Befehle wird dem Computer überlassen. Eine noch höhere Automatisierung ist mit dem Programm XSUB zu erreichen. Damit ist es möglich, die aufgerufenen Programme durch die Aufgabendatei zu bedienen.

4. ED Der Editor, ein unglaublich vielfältiges Programm, das beim Arbeiten mit CP/M oft gebraucht wird. Mit dem Editor ist es möglich, dem Computer Zeichen einzugeben, ohne daß diese als Befehl aufgefaßt und sofort ausgeführt werden. So ist beispielsweise eine einfache Textverarbeitung mit dem Editor möglich. Aber auch zur Erstellung, Veränderung und Korrektur von Programmen ist er einsetzbar.

5.SYSGEN ist ein Dienstprogramm mit dem sich das Betriebssystem auf jede formatierte Diskette überspielen läßt.

6. CONFIG dient der Zusammenstellung der Hardwarekonfiguration. Es kann gewählt werden zwischen den folgenden Einstellungsmöglichkeiten: Anzahl der Laufwerke, Druckertyp, Schriftmodus und Belegung der Funktionstasten.

7. DUMP zeigt den Inhalt einer Datei in den binären Werten seiner Bytes. (Hexdump).

8. PIP unterstützt den Datenaustausch mit den angeschlossenen Peripheriegeräten, wie dem Drucker, der Floppy und als Überbleibsel aus der Entstehungszeit von CP/M, mit Lochstreifenlesern und Fernschreibern. So können Dateien ausgedruckt oder kopiert werden; auch läßt sich aus mehreren Einzeldateien eine neue kreieren.

9. COPY ist ein speziell für Commodore hinzugefügtes Diskettenhilfsprogramm. Mit COPY lassen sich Disketten formatieren und kopieren. Der Backup einer gesamten Diskette dauert (auch mit zwei Laufwerken) immer noch zirka 15 Minuten.

10. ASM ist ein Assembler, mit dem sich in mnemonischer Schreibweise verfaßte Programme übersetzen lassen. Neben ASM wird hierzu auch das Programm LOAD verwendet, das die durch das Assemblieren entstandene HEX Datei in eine COM Datei umwandelt. Durch diesen Prozeß werden die Daten so aufbereitet, daß sie direkt aufrufbar sind und ausgeführt werden können.

Eigentlich eine ganze Menge, was eine Systemdiskette alles ermöglicht: einfache Textverarbeitung, Programmierung und Datenübertragung, wenn auch in einer unkomfortablen Form. Würde man vergleichbare Programme für den Commodore 64 kaufen, um mit ihm in seiner eigenen Sprache zu kommunizieren, müßte man wahrscheinlich mehr ausgeben als für das CP/M-Modul. So gesehen lohnt sich die Anschaffung auf jeden Fall.

(Arnd Wängler)

Pa

leistungsfähiger

Für den Commodore 64 gibt es bereits einige Versionen der Programmiersprache Pascal. Dieser Bericht stellt wichtige Elemente der Sprache vor und vergleicht sie mit äquivalenten Basic-Lösungen.

```
PROGRAM WURZEL(INPUT,OUTPUT);
VAR z : INTEGER;

BEGIN
  WRITE('? '); READLN(z);
  WHILE z >= 0 DO
    BEGIN
      WRITELN (SQRT(z));
      WRITE('? '); READLN(z);
    END
END.
```

Listing 1. Programm mit WHILE-Schleife

```
PROGRAM WURZEL(INPUT,OUTPUT);
VAR z : INTEGER;

BEGIN
  REPEAT
    WRITE('? '); READLN(z);
    IF z >= 0 THEN
       WRITELN (SQRT(z))
  UNTIL z < 0
END.
```

Listing 2. Programm mit REPEAT-Schleife

```
10 REM            PROGRAMM WURZEL
20 REM            VARIABLE Z%
30 INPUTZ%
40 PRINT Z%
50 IF Z% >= 0 THEN PRINT SQR(Z):GOTO 30
60 END
```

Listing 3. Basic-Programm mit derselben Leistung wie die Pascal-Programme aus Listing 1 und 2.

Wie lange besitzen Sie schon den Commodore 64? Wenn Sie ihn wie ich schon längere Zeit und einige Gehversuche in Basic schon hinter sich haben, die Phase der Spielsucht und des Programme-Scheffelns überstanden haben, dann werden Sie sich wohl wieder dem Programmieren in Basic zuwenden, nur diesmal ungleich intensiver. Jetzt stellt sich jedoch heraus: das eingebaute Basic hält den gewachsenen Anforderungen nicht stand.

Eine neue Sprache, gut, aber welche? Da gibt's ja eine Unmenge davon, zum Beispiel Forth, Cobol, Algol, Fortran, Pascal, Modula und Assembler, die nur »ehrfürchtig« genannt wird, da Assembler angeblich nur von absoluten Spitzenkönnern beherrscht wird. Zu den Könnern zählt man sich im allgemeinen noch nicht und läßt eben die Maschinensprache beiseite (weshalb eigentlich?) und sucht sich eine sogenannte »höhere« Programmiersprache. Die Wahl dürfte wahrscheinlich auf Pascal fallen, weil Forth zu fremd, Fortran zu antiquiert, Cobol zu »geschäftig«, Algol zu wissenschaftlich und Modula zu neu ist. Nun, Pascal hat schon einiges zu bieten, was Basic nicht hat. Die Schlagwörter sind:

— Strukturierte Programmierung
— Blockorientierte Programmierung
— Operatoren auf Mengen
— Lokale Parameter
— Möglichkeit für Rekursionen
— eigene Typendefinitionen
— Records etc.

Über Schlagworte läßt sich bekanntlich streiten, aber ich möchte beweisen, daß sie für Pascal wirklich zutreffen.

Niklaus Wirth, der diese Sprache aus den beiden Hauptlinien Algol 68 und Fortran beziehungsweise PL/1 1971 an der ETH Zürich entwickelte, hat sie Pascal genannt, zu Ehren des französischen Mathematikers und Philosophen Blaise Pascal (zirka 1650). Man kann deshalb von einem Wirth-Standard im Gegensatz zum UCSD-Pascal sprechen. Das UCSD-Pascal entstand in Kalifornien und enthält nun auch den Typ »STRING« sowie Operatoren auf diesen Typ wie man es von Basic her gewöhnt ist. Ansonsten ist der Unterschied nicht annähernd so groß, wie derjenige zwischen Basicdialekten.

Pascal ist wegen seiner Klarheit im Aufbau und seiner enormen Leistungsfähigkeit zur wichtigsten, wissenschaftlichen Programmiersprache geworden.

Ein äußerlicher Unterschied gegenüber Basic besteht darin, daß Pascal vollkommen formatfrei ist, das heißt es gibt keine Zeilennummerierung. Man darf daher eine Befehlssequenz (statements) über eine Bildschirmzeile hinaus auf der nächsten fortsetzen, denn, als gültigen Statementstrenner gibt es nur das Semikolon (;). Pascal kennt daher den Befehl »GOTO/GOSUB Zeilennummer« nicht. Aber dafür ist Pascal sehr stark blockorientiert. Ein Block ist im einfachsten Fall entweder ein Befehl oder eine mit »BEGIN« und »END« umschlossene Befehlssequenz. Andere Blocks sind »PROCEDURE«, eine Art Unterprogramm, und »FUNCTION«, auch eine Art Unterprogramm, aber mit dem Unterschied, daß man einer »FUNCTION« einen Wert zuweisen kann. Darauf komme ich später zurück. Ein Beispiel dazu:

FOR i := 10 DOWNTO 3 DO statement

Diesem Statement entspricht in Basic die folgende Sequenz:

FOR i = 10 TO 3 STEP —1: befehle: NEXT

Diese ganze Schleife ist ein Statement. Obwohl sie mehrere Befehlswörter enthält, muß man sie nicht mit »BEGIN« und »END« umklammern. Ebenfalls als ein Statement gilt eine Wertzuweisung der Form:

i := i + 1;

Der Operator ':=' weist der links davon stehenden Variablen den Wert des rechten Ausdrucks (expression) zu. Er ersetzt nicht den Vergleichsoperator '='. Diesen braucht man in Pascal korrekterweise nur für Vergleiche wie »IF i = 5 THEN....«

Wenn ich schon bei Vergleichsstatements bin, kann ich die drei ebenfalls »mehrwortigen« Statements besprechen, die Pascal für bedingte Abarbeitung eines Blocks besitzt, wobei eine Bedingung (condition) von der Form ist »variable =,⟨⟩,⟨=,=⟩,⟨ expression«:

— IF condition THEN statements ELSE statements
— WHILE condition DO statement
— REPEAT statements UNTIL condition

Das erste Statement existiert fast in identischer Form in Basic, außer der Tatsache, daß »ELSE« im Basic des Commodore 64 natürlich fehlt. Die unteren beiden haben einen Schleifencharakter. Wofür nun aber zwei Schleifen, die sich doch fast nicht unterscheiden? Nun, bei näherer Betrachtung finden Sie sehr wohl einen interessanten Unterschied: Die »WHILE«-Schleife testet eine Bedingung bevor eine Befehlssequenz ausgeführt wird (Solange wie...wiederhole...), die »REPEAT«-Schleife hingegen testet erst nach der Ausführung (Wiederhole...bis...). Sie werden vielleicht sagen, das sei doch nun wirklich nur Haarspalterei. Mitnichten! In den folgenden zwei Beispielen sehen Sie, wie man damit elegant iterative Probleme mit Bedingungen programmieren kann. Kümmern Sie sich vorerst nicht um



die anderen Statements, die in den beiden Progrämmchen sonst noch vorkommen, sondern konzentrieren Sie sich bitte nur auf die Schleifen.

Beide Programme (Listing 1 und Listing 2) berechnen die Quadratwurzel einer Zahl:

Die unterstrichenen Worte sind reservierte »Steuerworte« in Pascal, die nicht anders gebraucht werden dürfen. Die übrigen Worte sind Standardprozeduren. Man erkennt sie auch daran, daß sie ein oder mehrere Argumente in Klammern annehmen können. Ein Statement kann aus Steuerworten und Prozeduren bestehen, was man leicht erkennen kann, weil nach 'WHILE'...'DO' bekanntlich nur ein Statement folgen darf, denn sonst müßte man ein Ende der »WHILE«-Schleife definieren. »PROGRAM«... sind die Programmköpfe, »VAR«... die Deklarationsteile, auf die ich bald eingehen werde und innerhalb »BEGIN« und »END.« befindet sich das eigentliche Programm. »WRITE« beziehungsweise »WRITELN« entsprechen im Commodore-Basic »PRINT;« beziehungsweise »PRINT«, »READLN« dem »INPUT«. »SQRT« entspricht »SQR« und berechnet die Quadratwurzel.

Nun zur Routine selbst: Wie Sie wissen, kann man nur aus Zahlen >= 0 reelle Quadratwurzeln ziehen, deshalb muß man vor der Berechnung jeweils testen, ob die mit »READLN« gelesene Zahl >= 0 ist. Dies geht natürlich sehr elegant mit der »WHILE«-Schleife, die ja vor der Ausführung den Test durchführt. In der »REPEAT«-Schleife muß hingegen unbedingt ein Test mit »IF« gemacht werden. Weshalb wird aber die Eingabe zweimal geschrieben? Nun, erstens enthalten Variablen in Pascal, nachdem sie ins Leben gerufen wurden, keinen bestimmten Wert und zweitens muß der erstmalige Test in der »WHILE«-Schleife etwas »Wohldefiniertes« zum Testen haben. Man sieht auch gleich, daß der ganze Inhalt der »WHILE«-Schleife übersprungen wird, falls die erste Eingabe < 0 ist. Hätte man nun in der »WHILE«-Schleife keine Gelegenheit mehr, »z« zu verändern, käme man niemals mehr aus ihr heraus. In der »REPEAT«-Schleife verhält es sich diesbezüglich ähnlich. Nun noch schnell das äquivalente Basicprogramm:

In dem äquivalenten Basic-Programm (Listing 3) entsprechen der kleinen Standardprozedur »READLN« die Zeilen 30 und 40. Es ist sehr schön zu sehen, daß man in Basic nicht ohne das »GOTO« auskommt und daß das Programm jetzt schon schlechter ist, als dasjenige in Pascal. Stellen Sie sich schon jetzt mal vor, wie es sich erst mit längeren bis sehr langen Programmen verhält, bei denen man nicht so schnell sieht, was alles wiederholt wird, weil zwischen der angesprungenen Zeile und dem entsprechenden »GOTO« eventuell mehr als 100 Zeilen liegen.

Ein weiteres Pascal-Statement, von dem man sagen könnte, daß es irgend etwas teste, ist das »CASE«-Statement. Es lautet:

```
— CASE selektor OF
  marke 1 : STATEMENT;
  marke 2 : STATEMENT;
  marke 3 : STATEMENT;
  ........ : ........
  ........ : ........
  marke n : STATEMENT
END;
```

Mit ihm kann man eine aus mehreren Möglichkeiten auswählen, je nachdem welcher Marke der Selektor entspricht. Es entspricht den »ON x GOTO« und »ON x GOSUB«-Verteilern. Aber »CASE« ist dagegen ungleich vielseitiger. Bekanntlich darf »x« in Basic weder ein Ausdruck noch ein alphanumerisches Zeichen sein. In Pascal darf der Selektor ein beliebiges Zeichen sein. Listing 4 zeigt dies.

Je nachdem, welchen Wert »tagnr« hat, werden der Variablen »tag« die entsprechenden 3 Buchstaben zugewiesen. Stimmt der Selektor mit keiner der Marken überein, so ist im Standardpascal nicht definiert, was dann geschieht. In UCSD-Pascal hingegen wird einfach beim nächsten Statement fortgefahren. Hier folgt nun noch das kleine äquivalente Basicprogramm dazu (Listing 5). Da im Basic des Commodore 64 dem »IF...THEN«-Befehl das »ELSE« fehlt und weil eine Basiczeile nicht mehr als 80 Zeichen zuläßt, muß man auf die Konstruktion verschachtelter »IF...THEN...ELSE IF...THEN... ELSE IF...'s zur Emulation von »CASE« verzichten. Man sieht, daß man zur Emulation dieses Statements in Basic sehr redundant programmieren muß.

Es ist dies sicher nicht die einzige und schon gar nicht die eleganteste Lösung, aber wahrscheinlich dieje-

```
PROGRAM TAGRECHNUNG (INPUT,OUTPUT);
VAR   tag : (MON,DIE,MIT,DON,FRE,SAM,SON);
    tagnr : integer;
BEGIN
  (*                                                  *)
  (* HIER SOLLTE EINE RECHNUNG STEHEN, DIE EINE       *)
  (* ZAHL ZWISCHEN 0 UND 6 IN tagnr HINTERLEGT.       *)
  (*                                                  *)
  CASE tagnr OF
    0 : tag := MON;
    1 : tag := DIE;
    2 : tag := MIT;
    3 : tag := DON;
    4 : tag := FRE;
    5 : tag := SAM;
    6 : tag := SON
  END;  (* OF CASE *)
  WRITELN (tagnr,' entspricht ',tag)
END.
```

Listing 4. Der Selektor in der CASE-Anweisung kann eine Variable oder ein Ausdruck sein. Hier dient eine INTEGER-Variable als Selektor.

nige, die einem zuerst in den Sinn kommt. Eine weitere bevorzugte Verwendung von »CASE« ist die Menüsteuerung, wie Listing 6 zeigt.

Zu Listing 6 ist nicht viel zu sagen, außer daß man sieht, daß der Selektor (Variable »w«) auch ein alphanumerisches Zeichen sein kann und daß den Marken (»A«, »B«, »C«) mehrere Statements folgen können. Hier also Zuweisungen und Prozeduren. Hinzu kommt noch eine neue Standardprozedur genannt »PAGE«. Sie produziert einen Seitenvorschub auf einer Text-Ausgabedatei, weshalb dieser Prozedur ein Dateiname folgen muß. »PAGE (OUTPUT)« produziert also einen Seitenvorschub auf dem Bildschirm, der sich als Löschen des Schirms äußert.

Blockstruktur, lokale und globale Variablen

Nun komme ich endlich zu demjenigen Punkt, welchen ich bisher immer vor mir herschob, Ihnen aber schon vier mal in den Programmen vorgesetzt habe: Es ist dies der Programmkopf mit dem wichtigen Deklarationsteil. Die Blockstruktur ist ein Hauptmerkmal von Pascal. Man kann sich eine Ebene vorstellen, die bezeichnet wird mit »PROGRAM name«. In ihr sind überall und jederzeit alle Konstanten und Variablen verfügbar und änderbar, die in dem zu dieser Ebene gehörigen Deklarationsteil angegeben werden müssen. Ein Programm, das völlig ohne Prozeduren und Funktionen auskommt, liegt in dieser Ebene, und somit leben alle Variablen in ihm (das heißt sie sind gültig), eben weil sie im Deklarationsteil dieser Ebene stehen. Im Deklarationsteil, der zu dieser Ebene gehört, wird nun gesagt, ob und welche Konstanten, Typen und Variablen in dieser Ebene gebraucht werden. Man muß also alle in dieser Ebene verwendeten Variablen deklarieren. Wenn man nun ein Unterprogramm, genannt Prozedur, einführt, so bildet man eine neue Ebene, die jetzt auf derjenigen liegt, die »PROGRAM« genannt wurde (es entsteht mit der Zeit eine Art Relief, indem Ebene auf Ebene zu liegen kommt). Will man die Variablen aus der Hauptebene (»PROGRAM«) benützen, so kann man das bedenkenlos tun.

Man kann aber auch Konstanten, Typen und Variablen definieren, die nur in dieser und jeder darauffliegenden Ebene leben, das heißt man kann von der Hauptebene diese Variablen nicht ansprechen, weil sie für die Hauptebene nicht existieren, da sie nicht in deren Deklarationsteil verzeichnet sind. Daraus ergibt sich, daß man die gleichen Variablennamen verwenden darf, die eigentlich schon in der Hauptebene verteilt worden sind. Obwohl, gleich benannt, beeinflussen sie einander in keiner Weise. Beide Inhalte bleiben erhalten. Je nachdem, in welchem Block man sich befindet, sind gerade die Variablen von dessen Deklarationsteil aktiv. Man kann also, indem man in der Prozedur einen Deklarationsteil hat, eine Sperre aufbauen, die ein Verändern einer gleichlautenden Variable aus einer tieferliegenden Ebene verhindern

```
 10 REM       PROGRAMM TAGRECHNUNG
 20 REM       VARIABLEN tag$
 30 REM                 tagnr%
 40 REM       BERECHNUNG VON tagnr
 50 IF tagnr% = 0 THEN tag$ = "MON"
 60 IF tagnr% = 1 THEN tag$ = "DIE"
 70 IF tagnr% = 2 THEN tag$ = "MIT"
 80 IF tagnr% = 3 THEN tag$ = "DON"
 90 IF tagnr% = 4 THEN tag$ = "FRE"
100 IF tagnr% = 5 THEN tag$ = "SAM"
110 IF tagnr% = 6 THEN tag$ = "SON"
120 PRINT tagnr%" entspricht "tag$
130 END
```

Listing 5. Dem Pascal-Programm in Listing 4 entsprechendes Basic-Programm.

```
PROGRAM RAHMEN (INPUT,OUTPUT);
VAR     i,k,l : real;
          c,w : char;

 PROCEDURE GETDATA;
 BEGIN
  (*                              *)
  (*    HIER STEHT DIE DEFINITION *)
  (*                              *)
 END;    (* VON GETDATA *)

 PROCEDURE USEDATA;
 BEGIN
  (*                              *)
  (*    HIER STEHT DIE DEFINITION *)
  (*                              *)
 END;    (* VON USEDATA *)

 PROCEDURE STOREDATA;
 BEGIN
  (*                              *)
  (*    HIER STEHT DIE DEFINITION *)
  (*                              *)
 END;    (* VON STOREDATA *)


BEGIN          (* DES HAUPTPROGRAMMES *)
  REPEAT
    PAGE (OUTPUT);
    WRITELN; WRITELN; WRITELN;
    WRITELN (' BITTE WAEHLEN SIE: ');
    WRITELN;
    WRITELN (' A : DATEN HOLEN ');
    WRITELN (' B : DATEN BEARBEITEN ');
    WRITELN (' C : DATEN ABSPEICHERN ');
    WRITELN (' Q : BEENDEN ');
    READ (w);
    CASE w OF
     'A' :  BEGIN    i := i + 1.5; GETDATA;   END;
     'B' :  BEGIN    k := k - 2.9; USEDATA;   END;
     'C' :  BEGIN    l := l - 1;  STOREDATA;  END
    END  (* OF CASE *)
    WRITELN;
    WRITELN ('IST IHRE ARBEIT BEENDET     J/N ?');
    READ (c);
  UNTIL c = 'J'
END.          (* DES HAUPTPROGRAMMES *)
```

Listing 6. Menüsteuerung über CASE-Anweisung in Pascal (Programmauszug)

```
PROGRAM INITMATRIX (INPUT,OUTPUT);
CONST       n = 25;        (* ZEILEN *)
            m = 15         (* SPALTEN *)
TYPE  matrix = ARRAY [0..n,0..m] OF INTEGER;
VAR        a : matrix;
         k,i : INTEGER;

  PROCEDURE ZEILE;
  VAR          i : INTEGER;
  BEGIN
    FOR i := 0 TO m DO
      a[k,i] := 0;
  END;
BEGIN      (* DES HAUPTPROGRAMMES *)
  k := 0;
  FOR i := 0 TO n DO
    BEGIN
      ZEILE;
      k := k + 1;
    END;
    WRITELN ('DIE MATRIX IST JETZT INITIALISIERT WORDEN.');
END.      (* DES HAUPTROGRAMMES *)
```

Listing 7. Sowohl im Hauptprogramm als auch im Unterprogramm »Zeile« ist eine Variable i definiert worden. Beide Variablen, trotz demselben Namen, sind völlig unabhängig voneinander.

```
PROCEDURE OUT;
VAR  a,b : INTEGER;

 PROCEDURE IN (i :INTEGER; VAR j :INTEGER);
 BEGIN
   j := i * 2;
 END;

BEGIN
  .........
  .........
  IN (a,b);
  .........
  .........
END.
```

Listing 8. Parameterübergabe an Prozeduren. Es gibt formale und aktuelle Parameter.

kann. Aus dieser Tatsache entstand der Begriff der »Lokalen Variablen«. Ein typisches Anwendungsbeispiel dazu ist das »FOR i := x TO y DO«-Statement. Da man mit diesem Statement oft Indices verändert, heißt die Laufvariable meistens »i« für Index, und so hat es sich eingebürgert, daß man sie wenn möglich immer »i« nennt. Damit nun niemals irgendwelche Komplikationen auftreten können, deklariert man diese Variable »i« meistens in jeder Ebene und zwar vom Typ »INTEGER«. Ich hoffe, daß dies am nächsten Beispiel (Listing 7) ein bißchen klarer wird:

In Listing 7 gibt es zum ersten Mal einen fast ausgelasteten Deklarationsteil. Er ist strikt gegliedert und die Teile der Deklaration müssen immer in derselben Reihenfolge auftreten. Im Standardpascal lautet die Abfolge folgendermaßen:

— Labelteil	Zuweisung der Sprungmarken
— Konstantenteil	Deklaration und Zuweisung
— Typenteil	Deklaration und Definition
— Variablenteil	Deklaration und Typenzugehörigkeit
— Prozeduren und Funktionen	Deklaration und Definition

In UCSD-Pascal gibt es zwar den Labelteil, aber das »GOTO«-Statement nur bedingt. Man darf daher nicht ein Label anspringen, welches außerhalb des aktuellen Blockes liegt. Ein Aussteigen aus dem gerade bearbeiteten Block ist nur mit »EXIT procedure p« möglich. Der aktuelle Block wird verlassen und die Prozedur »p« wird ausgeführt. Eine Labeldeklaration geschähe dann wie folgt:
LABEL 0,27,56,876,9999
Damit hätte ich 5 Labels deklariert, die irgendwo im betreffenden Block stehen können. Die Zahlen haben nichts mit irgendwelchen Zeilennummern zu tun (man kann sie natürlich so verwenden).

Die Konstantendeklaration:
CONST name = wert;
Die Typendeklaration:
TYPE name = Definition des Typs;
Die Variablendeklaration:
VAR name : Typenzugehörigkeit;
Die Prozeduren- und Funktionendeklaration:
PROCEDURE name (Übergabeparameter);
Definition der Prozedur
FUNCTION name (Übergabeparameter);
Definition der Funktion

Nun ist zwar endlich klar, was ein Deklarationsteil ist, aber die Typenarten, von denen ich schon zwei in den Beispielprogrämmchen benutzt habe, sind noch immer nicht klar. Nun, es gibt einmal vordefinierte Typen, diese sind:

INTEGER	Bereich der ganzen Zahlen von −32768 bis +32767
REAL	Bereich der reellen Zahlen von zirka 1E-38 bis zirka 1E38
BOOLEAN	Nur 2 Werte, 0 = false, 1 = true
CHAR	1 Zeichen, das dem ASCII-Code von 32 bis 127 entspricht
TEXT	Abkürzung für »file of char«, siehe weiter unten
Im UCSD-Pascal existiert noch der Typ:	
STRING	Zeichenkette, array of char

Eine ebenfalls vordefinierte Abänderung der Standardtypen erhält man, indem man schreibt: »FILE OF standard typ«. So deklariert man eine sequentielle Datei, deren Elemente einem der Standardtypen angehören. Eine ungeheure Flexibilität erreicht Pascal nun, indem man eigene Typen definieren kann. Im letzten Beispiel habe ich auch einen eigenen Typen definiert, nämlich den Typ »matrix«. Er ist definiert als ein zweidimensionales Feld der Größe »m« * »n« Elemente des Standardtyps »INTEGER«. In der Typendeklaration hat man fast unbeschränkte Möglichkeiten der Typengenerierung. Da nun der Typ definiert ist, darf ich ihn im Variablendeklarationsteil verwenden. Auch das habe ich gemacht: Es sei »a« vom Typ »matrix«. Im Konstantenteil hat man nur die Standardtypen zur Auswahl, da die Typendeklaration erst später folgt. Einen großen Nachteil hat die Sache mit dem Deklarationsteil: Da man nur Variablen benutzen darf, die man deklariert hat, kann man die Reservation von Speicherplatz für die Variablen nicht optimieren. Man sieht das sehr deutlich an der Variablen »a«. Die Größe der Matrix ist bestimmt durch die Konstanten »m« und »n«, die ich aber schon vorher deklarieren mußte. Ob ich nun wirklich die ganze Größe voll ausnutze oder nicht, ich muß die Array-Grenzen angeben. Unter Umständen verschleudere ich sehr viel Speicherplatz, eventuell reicht er aber nicht einmal. In Basic dagegen kann man schreiben:
10 INPUT "WIEVIELE ZEILEN, SPALTEN";m,n
20 DIM matrix(m,n)

In Pascal geht es nicht so bequem, aber es ist möglich. Es werden noch einige Beispiele folgen, in denen an-

dere Typen gebraucht werden. Das Beispiel (Listing 7) zeigt, daß neben- und nacheinander zwei Variablen »i« leben können, die einander nicht zerstören. Die eine ist global, weil sie im Programmkopf deklariert worden ist, die andere ist lokal, da sie erst in der Prozedur deklariert wird. Man muß aber klar erkennen, daß man zu keinem Zeitpunkt die Werte beider Variablen zugleich ausgeben kann. Zu den Prozeduren ist zu sagen: Jede Prozedur hat einen Namen. Man kann ihr Werte übergeben, indem man hinter dem Namen eine Klammer öffnet und die Variablennamen sowie deren Typenzugehörigkeit eingibt. Man kann nun aber auf zwei Arten Werte übergeben. Dies soll am Beispiel zweier Prozeduren (Listing 8) gezeigt werden:

Man erkennt sofort, daß an die Prozedur »IN« zwei Werte übergeben werden, fragt sich nur, wo der Unterschied der zwei Übergabearten liegt. Dies ist schnell geklärt:

Die Parameter in Klammern bei der Prozedur »IN (i: INTEGER; VAR j: INTEGER)« sind sogenannte »formale Parameter«, diejenigen aber im Prozedurenaufruf »IN(a,b)« nennt man »aktuelle Parameter«. Die formalen Parameter zeigen an, daß (hier zwei) Parameter übergeben werden, die in der betreffenden Prozedur »i« und »j« heißen werden. Die aktuellen Parameter dagegen beinhalten die Werte, die tatsächlich übergeben werden. In unserem Beispiel heißt das, daß die aktuellen Parameter »a« beziehungsweise »b« den formalen Parametern »i« beziehungsweise »j« entsprechen. Nun, einen kleinen Unterschied gibt es aber doch, wie die Notation der formalen Parameter vermuten läßt:

```
PROGRAM VERGLEICH (INPUT,OUTPUT);
CONST            max = ....;
TYPE   zahlenmenge = ARRAY [0..max] OF BOOLEAN;

FUNCTION M1INM2 (m1,m2 :zahlenmenge) :BOOLEAN;
VAR          i : INTEGER;
          flag : BOOLEAN;

BEGIN
  flag := TRUE; i := 1;
  WHILE flag AND i <= max DO
    BEGIN
     IF m1[i] AND m2[i]
        THEN flag := TRUE
        ELSE flag := FALSE;
     i := i + 1;
    END;
END;          (* VOM M1INM2 *)

BEGIN
  (*                                              *)
  (* HIER FOLGT NUN EIN HAUPTPROGRAMM, DAS ZWEI   *)
  (* MENGEN ENTSPRECHEND DER DEKLARATION ERZEUGT, *)
  (* DIE NUN VERGLICHEN WERDEN MUESSEN MIT M1INM2 *)
  (*                                              *)
END.
```

Listing 9. Es wird geprüft, ob eine Menge in einer anderen Menge enthalten ist.

Dem formalen Parameter »j« wird der Inhalt des aktuellen Parameters »a« übergeben.

Dem formalen Parameter »j« jedoch wird die Speicherplatzadresse des aktuellen Parameters »b« übergeben. Ein kleiner aber wichtiger Unterschied.

Wenn nämlich die Prozedur »IN (i: INTEGER; VAR j: INTEGER)« ausgeführt wird, wird der Inhalt von »a« in die Variable »i« kopiert. Der Inhalt von »a« wird in dieser Prozedur nicht verändert. Dagegen wird beim Aufruf von »IN (i: INTEGER; VAR j: INTEGER)« nicht der Inhalt von »b« nach »j«, sondern deren Adresse zur Adresse von »j« kopiert. Das hat eine konsequenzenreiche Auswirkung: Alle Veränderungen von »j« in der Prozedur »IN« beeinflussen auch den Inhalt der globalen Variablen »b«, weil »j« nun ja einfach nur ein anderer Name für die Variable »b« ist. Die erste Methode der Parameterübergabe nennt man daher »call by value«, die zweite hingegen »call by reference«. Die Variablen sind nun in der betreffenden Prozedur sowie jeder weiteren darin verschachtelten Prozedur lokal bekannt. Man hat nur darauf zu achten, daß man bei der Übergabe keinen Typenkonflikt verursacht und daß beim Aufruf ebenso viele Parameter stehen wie beim Prozedurenkopf.

Diese Art der Parameterübergabe eignet sich besonders dazu, eigene Programmodule für eine Programmroutinenbibliothek bereitzustellen. Ein weiteres Beispiel dazu mit einer »FUNKTION«, für eine ganz bestimmte Anwendung, nämlich einer Vergleichsfunktion:

Gegeben sei folgende Deklaration:
CONST max = 10000;
TYPE zahlenmenge = ARRAY (1..max) OF BOOLEAN;

Eine Variable vom Typ »zahlenmenge« kann man auffassen als die Darstellung einer Menge von Zahlen zwischen »1« und »max«, wobei für jede solche Zahl der entsprechende Boolean-Wert im Array angibt, ob sie Element der Menge ist oder nicht (das heißt ob der korrespondierende Wert im Array »1« oder »0« ist).

Gesucht ist nun eine Pascal Funktion, die das Enthaltensein einer Menge »m1« in einer Menge »m2« (beide vom obigen Typ) prüft. Eine mögliche Lösung zeigt Listing 9.

Diese Funktion übernimmt nun aus dem Umfeld der Funktion zwei Variablen vom Typ »zahlenmenge« und weist sie den formalen Parametern »m1« und »m2« mittels der »call by value«-Methode zu. Neu bei der Funktion ist nun, daß hinter der Klammer noch deklariert werden muß, welchen Typs das Resultat der Funktion sein wird. Die Funktion ist ein wichtiger Bestandteil von Pascal. Ihr entspricht in Basic teilweise der Befehl »DEF FN name (variable) = expression«, außer daß er in Basic nur eine mathematische Formel definieren und nur eine numerische Variable übernehmen kann. Man kann mit ihm keinerlei Operationen definieren, die als Resultat einen String oder einen Charakter ausgeben.

Der zweite Teil erscheint in der nächsten Ausgabe und beschreibt die Programmierung mit Funktionen, dem Mengentyp, Aufzählungs- und Auszähltyp sowie die Definition von Datensätzen und den Einsatz von Zeigern (POINTER). Kurz erläutert werden Befehle zur Dateibearbeitung. Den Abschluß bildet eine kritische Auseinandersetzung mit vier Pascal-Versionen, die alle auf dem Commodore 64 lauffähig sind.

(Martin Baur)

Debugging — Fehlersuche in Basic-Programmen

Diese Situation kennt wohl jeder Besitzer eines Homecomputers zur Genüge: Da tippt man in stundenlanger Arbeit ein ellenlanges Listing ein, lehnt sich nach dem letzten »RETURN« einen Augenblick erleichtert zurück, gibt das magische Wort »RUN« ein — und natürlich läuft das Programm in keiner Weise so, wie es eigentlich sollte. Das Spektrum der möglichen Ereignisse reicht dabei vom simplen »SYNTAX ERROR« bis zum völligen Absturz des Programms.

Solange der Computer noch brav seine Fehlermeldungen ausgibt, hat man ja noch Glück gehabt. Kritisch wird die Situation dann, wenn auf dem Bildschirm ein eigenartiges Gemisch undefinierbarer Zeichen erscheint und sich der Computer weder durch Betätigen aller erreichbaren Tasten, noch durch gutes Zureden wieder auf den Boden der Tatsachen zurückholen läßt. Wenn man beim Eintippen eines Programms einmal an diesem Punkt angelangt ist, wird es Zeit, sich die erste Regel gut einzuprägen: Jedes Programm sollte vor dem Start unbedingt mit SAVE gesichert werden.

Mit dem SAVEn allein ist es allerdings noch nicht getan, es muß effektiv noch etwas gegen die im Programm enthaltenen Fehler unternommen werden. Diesen Vorgang bezeichnet man auch als »Debugging«. Das Wort ist von der englischen Bezeichnung »Bug« abgeleitet und bedeutet eigentlich »entwanzen«, wobei mit den »Wanzen« die Fehler gemeint sind, die sich überall im Programm verstecken. In der amerikanischen Umgangssprache hat sich das Wort ganz allgemein für das Suchen versteckter Fehler eingebürgert.

Ganz grob kann man zwischen zwei Arten von Fehlern unterscheiden. Einerseits gibt es die logischen Fehler, die mit schöner Regelmäßigkeit in der Entwicklungsphase eines Programms auftauchen, weil man dem Computer noch nicht genau genug gesagt hat, was er denn nun eigentlich machen soll. Diese Art von Fehlern erkennt man zumeist daran, daß das Programm anstandslos läuft, aber nicht immer die gewünschten Ergebnisse produziert. Die zweite Art von Fehlern ist wesentlich profanerer Natur und tritt praktisch jedesmal dann auf, wenn man ein Programm von einem fremden Listing oder auch von den eigenen Aufzeichnungen abtippt: Es handelt sich dann zumeist um schlichte Tippfehler oder um Fehler, die auf schlechter Lesbarkeit der Vorlage beruhen. Wir wollen uns im folgenden nur mit der zweiten Art von Fehlern beschäftigen.

Der Computer hilft bei der Fehlersuche

Wie geht man nun zweckmäßig vor, um alle Fehler zu finden, ohne das gesamte Programm von Anfang bis Ende mit der Vorlage vergleichen zu müssen? Nun, eine allgemeingültige Methode, die für alle Arten von Programmen anwendbar wäre, gibt es leider nicht. Dennoch erscheint ein gewisses systematisches Vorgehen durchaus angebracht.

Zunächst sollten wir uns darüber klarwerden, inwieweit uns der Computer selbst bei der Suche nach Fehlern helfen kann. Als erstes kommen einem dabei natürlich die Fehlermeldungen in den Sinn, die beim Commodore-Basic ja erfreulicherweise im Klartext erfolgen und recht vielfältig sind. Wer mit den englischen Bezeichnungen nicht sofort etwas anfangen kann, hat die Möglichkeit, die deutschen Erläuterungen dazu im Handbuch nachzuschlagen.

Was aber soll man tun, wenn der Computer gar keine Fehlermeldung ausgibt, sondern sich nach »RUN« einfach sang- und klanglos verabschiedet und auf keine Tasten mehr reagiert?

Nun, für solche Fälle gibt es im Basic des C 64 beziehungsweise des VC 20 zwei spezielle Befehle, die man immer dann in nicht zu geringem Umfang einsetzen sollte, wenn man nicht genau weiß, wo denn nun der Fehler steckt. Gemeint sind die Basic-Befehle STOP und CONT. Wenn der Computer beim Abarbeiten des Programms auf den Befehl STOP stößt, unterbricht er die Programmausführung und gibt eine Meldung »BREAK IN nnn« aus, wobei nnn die Zeilennummer ist, in der er die STOP-Anweisung gefunden hat. Alle Variablen und auch der Stackpointer bleiben dabei erhalten, so daß die STOP-Anweisung auch in Unterprogrammen und innerhalb von FOR-NEXT-Schleifen auftreten kann.

Nach einem solchermaßen erzwungenen Programmstop kann man sich im Direktmodus mit dem PRINT-Befehl über die Werte wichtiger Variablen informieren und sogar mit LIST einzelne Programmteile anschauen. Danach gibt man den CONT-Befehl, und das Programm wird ganz normal fortgesetzt. Wenn es zu Testzwecken notwendig erscheint, kann man während eines Stops auch im Direktmodus Varia-

blenwerte verändern oder FOR-NEXT-Schleifen verwenden. Allerdings dürfen weder neue Programmzeilen eingegeben noch alte gelöscht oder verändert werden, da dadurch gleichzeitig alle Variablen gelöscht werden und CONT danach nicht mehr möglich ist.

Es ist empfehlenswert, an kritischen Stellen im Programm STOP-Befehle einzufügen, um dadurch den Programmlauf verfolgen zu können und den Fehler immer mehr einzugrenzen. Kritische Stellen sind generell und ohne Ausnahme alle SYS- und USR-Aufrufe, desgleichen alle POKE-Befehle, über deren Bedeutung man sich nicht hundertprozentig im klaren ist. Im Zweifelsfalle sollte man auch nach jedem GOSUB im Programm zunächst einen STOP-Befehl einbauen, um sicherzugehen, daß das Unterprogramm auch wieder auf normalem Wege verlassen wird.

Fehlersuche mit STOP und PRINT

Jedesmal, wenn man die Harmlosigkeit, zum Beispiel eines SYS-Befehls, durch davor und danach plazierte STOP-Befehle festgestellt hat, kann man die STOPs natürlich wieder entfernen, um einen flüssigeren Programmablauf zu erreichen. Es empfiehlt sich, alle eingefügten STOP-Befehle auf einem Zettel zu notieren, um die Übersicht zu behalten. Schließlich dienen diese Befehle nur der Fehlersuche und müssen irgendwann einmal alle wieder entfernt werden.

In vielen Fällen kann man STOP-Befehle durch einfache PRINT-Anweisungen ersetzen. Das hat den Vorteil, daß keine Programmunterbrechung stattfindet und man nicht jedesmal CONT eintippen muß. Außerdem kann man in PRINT-Anweisungen auch zusätzliche Informationen geben, zum Beispiel Variablenwerte ausdrucken oder direkt auf ein spezielles Problem aufmerksam machen. Diese PRINT-Anweisungen sollten aber in irgendeiner Weise von den normalen Bildschirmausgaben unterschieden sein. Zum Beispiel kann man jede PRINT-Anweisung zur Fehlersuche mit fünf Sternchen oder fünf Pluszeichen beginnen lassen.

Will man sich mehrere Variable während des Programmlaufs ausdrucken lassen, sind kleine Unterprogramme recht hilfreich, die in einen freien Zeilenbereich geschrieben werden und die alle benötigten Ausgaben durchführen. Am Ende solcher Unterprogramme sollte eine GET-Schleife stehen, die das Programm auf Tastendruck weiterlaufen läßt. Statt langer PRINT-Listen braucht man so nur einen GOSUB-Aufruf überall dort im Programm einzufügen, wo dies sinnvoll erscheint.

Komfortables Debugging mit Basic-Erweiterungen

Das Arbeiten mit STOP und CONT mag manchem Computerneuling etwas ungewohnt erscheinen, aber es ist jedenfalls ein recht sicheres Mittel, einem immer wieder abstürzenden Programm auf die Schliche zu kommen.

Leider sind STOP und CONT auch schon die einzigen speziellen Debug-Befehle im Commodore-Basic. Viele Spracherweiterungen, wie zum Beispiel das bekannte Simons Basic oder Exbasic Level II, stellen zusätzliche Funktionen zur schnellen Fehlersuche zur Verfügung. Wichtige derartige Befehle sind zum Beispiel

* **TRACE** — listet während des Programmlaufs die gerade bearbeiteten Programmzeilen oder zumindest die Zeilennummern auf.

* **DUMP** — gibt eine Liste aller Variablen mit ihren derzeitigen Werten aus.

* **FIND** — sucht im Direktmodus eine Zeichenfolge im Programm.

* **ON ERROR GOTO ...** — ermöglicht die Fehlerbehandlung im Programm selbst (ohne Abbruch).

Wenn der Computer jedoch noch in manierlicher Weise seine Fehlermeldungen ausgibt, sind aufwendige Verfahren zumeist nicht nötig, denn zusammen mit der Meldung über die Art des Fehlers erfährt man ja auch die Zeilennummer des Auftretens.

Syntax Error: Wenn der Computer nur noch »Bahnhof« versteht

Die häufigste Fehlermeldung ist sicherlich der ungeliebte »SYNTAX ERROR«. Böse Zungen behaupten allerdings, beim VC 20 wäre es der »OUT OF MEMORY ERROR«. Wie dem auch sei, solange der Computer nur Syntax-Fehler meldet, kann man noch von Glück reden. Es handelt sich dabei meistens um einfache Tippfehler, die nach auflisten der entsprechenden Zeile leicht zu finden und zu korrigieren sind. Beliebte Fehler sind zum Beispiel fehlende oder überzählige Klammern und die Verwechselung ähnlicher Zeichen, wie zum Beispiel »0« und »O«, »l« und »I« oder »8« und »B«. Sehr häufig ist auch die Verwechslung von Punkt und Komma, was sich besonders in DATA-Zeilen verhängnisvoll auswirken kann, wie wir nachher noch sehen werden. Wenn Sie also irgendwo einen »SYNTAX ERROR« gemeldet bekommen und in der fraglichen Zeile auf Anhieb keinen Fehler finden, dann gehen Sie zuerst die vorhin genannten Punkte durch.

Eine gute Hilfe ist es, mit dem Cursor die fehlerhafte Zeile Zeichen für Zeichen abzufahren und dabei mit der Vorlage zu vergleichen. Kommen in der Fehlerzeile viele Klammern vor, dann empfiehlt sich häufig das Anlegen zweier Strichlisten für öffnende und schließende Klammern. Die Anzahlen müssen innerhalb jeder Basic-Anweisung übereinstimmen. Aber bitte keine Klammern mitzählen, die in Anführungszeichen stehen; diese haben mit der Syntax nichts zu tun.

Syntax-Fehler, die man nicht sehen kann

Ab und zu kann es vorkommen, daß man bei aller Sorgfalt einen Syntaxfehler nicht findet, wie zum Beispiel in der folgenden Basic-Zeile:

```
10 OPEN 1,4:PRINT#1,"HALLO":
CLOSE 1
```

Wenn der Computer hier dennoch einen Syntaxfehler meldet, dann kann das nur eine Ursache haben: Bei der Eingabe dieser Zeile wurden die Basic-Befehle in der bekannten Art und Weise abgekürzt. Der zweite Befehl wurde also als »?#1« eingegeben, was beim Auflisten wieder zu »PRINT#1« wird. Leider sind PRINT und PRINT# zwei völlig verschiedene Befehle, genauso wie INPUT und INPUT# oder GET und GET#. Alle diese #-Befehle darf man daher nie abkürzen. Die normalen PRINT-, INPUT- und GET-Routinen wissen nämlich mit dem nachfolgenden »#«

nichts anzufangen und es kommt zu einer Fehlermeldung.

Eine ähnliche Situation kann sehr leicht bei Verwendung langer Variablennamen auftreten. In einem Spielprogramm »Schiffe versenken« kann zum Beispiel die folgende Zeile auftreten:

10 ANZAHLSCHIFFE = 12

In dieser Zeile wird unweigerlich ein »SYNTAX ERROR« auftreten, weil der Computer innerhalb des Variablennamens »ANZAHLSCHIFFE« das Basic-Schlüsselwort »IF« entdeckt und sich bei aller Anstrengung nicht erklären kann, was eine IF-Abfrage an dieser Stelle soll. Trotz aller Vorteile für die Übersichtlichkeit eines Programms sei daher an dieser Stelle von der Benutzung langer Variablennamen abgeraten.

Was tun bei »OUT OF DATA«?

Eine andere häufig auftretende Fehlermeldung ist vor allem bei Anfängern gefürchtet, nämlich der »OUT OF DATA ERROR«. Gefürchtet ist dieser Fehler vor allem deswegen, weil die Zeilennummer, die der Computer zu dieser Fehlermeldung ausgibt, in den allermeisten Fällen keinen Hinweis darauf gibt, an welcher Stelle denn nun ein Fehler vorliegt. Listet man nämlich die fehlerhafte Zeile am Bildschirm auf, so findet man dort nur den READ-Befehl, für den keine DATAs mehr vorhanden waren. In der Regel steht dieser READ-Befehl innerhalb einer FOR-NEXT-Schleife. Ein typisches, wenn auch stark vereinfachtes Beispiel für das Auftreten von Fehlern im Zusammenhang mit DATA-Anweisungen ist in Listing 1 auf Seite 50 gegeben. In den Zeilen 100 bis 170 wird eine Prüfsumme über den ersten DATA-Block gebildet, und nur dann, wenn diese Prüfsumme in Ordnung ist, wird in den nächsten Programmteil verzweigt, wo aus dem zweiten DATA-Block Zahlen gelesen und an den Anfang des Bildschirms gePOKEt werden und dort das Wort »COMMODORE« bilden sollen.

Das Programm enthält nun einige Fehler in den DATA-Zeilen, die wir gemeinsam herausfinden wollen. Stellen wir uns einfach vor, wir hätten das Programm in Listing 1 aus einer Zeitschrift abgetippt und dabei einige Fehler in den DATA-Zeilen fabriziert. In Listing 2 sind zum Vergleich noch einmal die entsprechenden DATA-Zeilen des »Original«-Listings abgedruckt. Die Fehlersuche scheint somit recht einfach: Man vergleicht die paar DATAs in beiden Listings und wird dann schon den Fehler finden. Bei diesem kurzen Testprogramm stimmt das natürlich auch. Aber stellen wir uns doch einmal vor, daß die beiden DATA-Blöcke insgesamt vielleicht über drei volle Listing-Seiten gehen und nicht nur über drei Zeilen wie in unserem Beispiel. Dann lohnt es sich nämlich mit Sicherheit schon, wenn man etwas systematischer an die Fehlersuche herangeht.

Zuerst starten wir unser Programm nach Listing 1 einmal ganz arglos mit RUN, nachdem wir es vorher auf Kassette oder Diskette abgespeichert hatten. Der Programmverlauf ist zu Anfang ganz wie erwartet: Der Bildschirm wird gelöscht, es erscheint die Meldung »S = 270« und darunter »OK«, dann jedoch erscheinen am oberen Bildschirmrand statt eines längeren Wortes nur die beiden Zeichen »@« und »C« und das Programm bricht mit der Meldung »? ILLEGAL QUANTITY ERROR IN 230« ab. Was ist hier geschehen?

Wenn wir uns Zeile 230 einmal auflisten lassen, dann sehen wir

230 POKE B + I,X

Da wir wissen, daß nur Zahlen zwischen 0 und 255 gePOKEt werden können, vermuten wir den Fehler beim Wert der Variablen X. Um unsere Vermutung zu bestätigen, fragen wir den Computer doch einmal ganz einfach nach dem Wert von X, indem wir eintippen

PRINT X

und danach die RETURN-Taste betätigen. Wir erhalten als Antwort den Wert 1513, der tatsächlich zu groß ist, um in eine Speicherzelle zu passen. Wir vergleichen den zweiten DATA-Block mit dem Original (Listing 2) und stellen fest, daß wir bei der Eingabe das Komma zwischen den beiden Zahlen 15 und 13 in Zeile 360 vergessen haben. Das ändern wir, indem wir das Komma nachträglich einfügen. Dank dieses schnellen Erfolges bessert sich unsere Stimmung um einiges, was sich jedoch nach dem nächsten RUN sehr schnell wieder ändert. Zwar erscheint am Bildschirm zunächst ganz ordentlich die Prüfsumme des ersten DATA-Blocks und das dazugehörige »OK«, aber am oberen Bildschirmrand stimmt einiges noch nicht: Man liest dort die Zeichenfolge »@COMMOORE« statt »COMMODORE«.

Auf den ersten Blick würde man vielleicht vermuten, daß der Fehler nur im zweiten DATA-Block stehen kann, weil das Lesen des ersten Blocks keine Fehlermeldung erzeugt und sogar die Prüfsummenbildung stimmt. Diese Überlegung ist aber nicht ganz schlüssig. Denn durch Bildung einer einfachen Prüfsumme werden Vertauschungsfehler und überflüssige oder fehlende Nullen nicht erkannt. Die fünf DATA-Zeilen in Listing 3 ergeben zum Beispiel alle die gleiche Prüfsumme.

Man sollte sich also nie blindlings auf Prüfsummen verlassen. Sie sind zwar oft nützlich, um Fehler in DATA-Zeilen festzustellen, man darf aber aus der Richtigkeit der Prüfsummenprobe niemals auf die Abwesenheit von Fehlern schließen. Außerdem taucht eine weitere Schwierigkeit auf: Wenn man nur eine globale Prüfsumme über alle DATAs bildet, dann kann man zwar unter Umständen einen Fehler nachweisen, weiß aber immer noch nicht, wo er steckt. Da muß man dann schon zu anderen Mitteln greifen.

Mit Listing, Lupe und Logik« Dem Fehler auf der Spur

Um den Fehler aufzuspüren, können wir dem Computer einen großen Teil der Arbeit überlassen. Als erstes wollen wir feststellen, in welchem der beiden DATA-Blöcke der Fehler liegen könnte. Dazu veranlassen wir den Computer einfach, nur den ersten DATA-Block zu lesen, indem wir eine STOP-Anweisung hinter die erste FOR-NEXT-Schleife plazieren. Wir fügen also folgende Zeile ins Programm ein:

145 STOP

Wenn wir das Programm jetzt starten, erhalten wir die Meldung »BREAK IN 145«, die von unserem STOP-Befehl herrührt. Da aber das Programm bis Zeile 145 durchlaufen wurde, muß der erste DATA-Block an dieser Stelle vollständig gelesen worden sein. Die Variable X enthält natürlich immer noch den zuletzt gelesenen DATA-Wert. Wenn dieser Programmteil richtig gearbeitet hat, dann müßte X jetzt den Wert Null haben, denn dies ist ja gerade der letzte DATA-Wert aus Block 1, wie man anhand von Listing 1 oder 2 unschwer erkennen kann. Das können wir einfach nachprüfen, indem wir den Computer nach dem Wert von X fragen:

PRINT X

Zu unserem Erstaunen ist die Antwort aber nicht 0, sondern 12. Wir

werfen wieder einen Blick auf Listing 1 und stellen fest, daß die Zahl 12 die vorletzte Zahl im ersten DATA-Block ist. Offenbar wurde eine Zahl zuwenig gelesen! Es ist nun verlockend, einfach den Endwert der ersten FOR-NEXT-Schleife um eins zu erhöhen, um alle Werte des ersten Blocks zu lesen. Doch halt, hier ist Vorsicht geboten. Viel wahrscheinlicher als ein Fehler in einer FOR-NEXT-Schleife ist ein Fehler innerhalb der DATA-Zeilen. Bei so vielen Zahleneingaben kann man sich schließlich leicht mal vertippen. Betrachten wir das Problem also einmal von der anderen Seite. Wenn die Anzahl der gelesenen X-Werte stimmt, das Programm aber trotzdem nur bis zum vorletzten DATA-Wert kommt, dann enthält Block 1 vielleicht einen DATA-Wert zuviel. Wir wollen also die DATAs in Block 1 ganz gezielt überprüfen. Dazu schreiben wir in einen freien Zeilenbereich, zum Beispiel ab Zeile 1000, das folgende kleine Unterprogramm:

```
1000 PRINT "I = " ; I , "X = " ; X
1010 GET A$ : IF A$ < > CHR$(32)
THEN 1010
1020 RETURN
```

In die erste FOR-NEXT-Schleife fügen wir direkt hinter die READ-Anweisung einen Aufruf dieses Unterprogramms ein:

```
125 GOSUB 1000
```

Wenn wir das Programm nun laufen lassen, geschieht folgendes: Der Computer gelangt mit Zeile 110 in die Leseschleife. In Zeile 120 wird jeweils ein DATA-Element gelesen. Dann erfolgt mit der eingefügten Zeile 125 ein Sprung in das vorhin geschriebene Unterprogramm. Dieses Unterprogramm druckt den Wert der Zählvariablen I und den soeben gelesenen DATA-Wert X aus und wartet dann, bis die Leertaste betätigt wird. Dann kehrt das Unterprogramm zurück und die Schleife wird nach dem NEXT in Zeile 140 erneut durchlaufen. Auf diese Art und Weise erhält man am Bildschirm eine übersichtliche Darstellung der gelesenen DATA-Werte, die man leicht mit der Vorlage vergleichen kann. Wenn wir das Programm jetzt starten, erhalten wir jeweils nach Drücken der Leertaste eine Bildschirmanzeige, etwa in der folgenden Art:

```
I = 1  X = 12
I = 2  X = 33
I = 3  X = 11
```

und so weiter bis schließlich das Ende von DATA-Zeile 310 erreicht wird:

```
I = 9   X = 18
I = 10  X =  0
```

Nanu? Das hatten wir eigentlich nicht erwartet. X = 18 ist der letzte Wert in Zeile 310, und danach sollte eigenlich der erste Wert aus der nächsten DATA-Zeile gelesen werden, nämlich X = 11. Woher also kommt dieser Wert Null bei I = 10? Ein Vergleich von Zeile 310 in Listing 1 (abgetippt) mit Listing 2 (Original) führt uns auf des Rätsels Lösung. Offenbar haben wir beim Abtippen am Ende von Zeile 310 noch ein Komma gesetzt, was da nicht hingehört. Ein Komma in einer DATA-Anweisung trennt für unseren Commodore-Computer aber immer zwei Werte voneinander, und da er hinter dem letzten Komma nichts mehr findet, setzt er kurzerhand den Wert Null dafür an.

Damit haben wir den überzähligen DATA-Wert im ersten Block gefunden. Wir entfernen das Komma in Zeile 310, löschen die Zeile 125 mit dem GOSUB-Befehl und ebenso die Zeile 145 mit dem nun nicht mehr benötigten STOP-Befehl.

Ein erneuter Probelauf des Programms schreibt die Zeichenfolge »COMMOORE« links oben in den Bildschirm — und bringt die Fehlermeldung »OUT OF DATA ERROR IN 220«. Nach Auflisten der Zeile 220 sehen wir leider nur

```
220 READ X
```

Das bringt uns nicht viel weiter. Die Fehlermeldung und das verstümmelte Wort »COMMOORE« am oberen Bildschirmrand deuten aber auf einen fehlenden DATA-Wert in Block 2 hin. Untersuchen wir also Block 2 einmal genauer. Das Unterprogramm zum Ausdrucken der Werte von I und X am Bildschirm befindet sich ja ab Zeile 1000 noch im Speicher. Wir brauchen daher nur einen entsprechenden GOSUB-Befehl in die zweite Leseschleife einzufügen, am besten gleich nach dem READ-Befehl, also etwa in Zeile 225:

```
225 GOSUB 1000
```

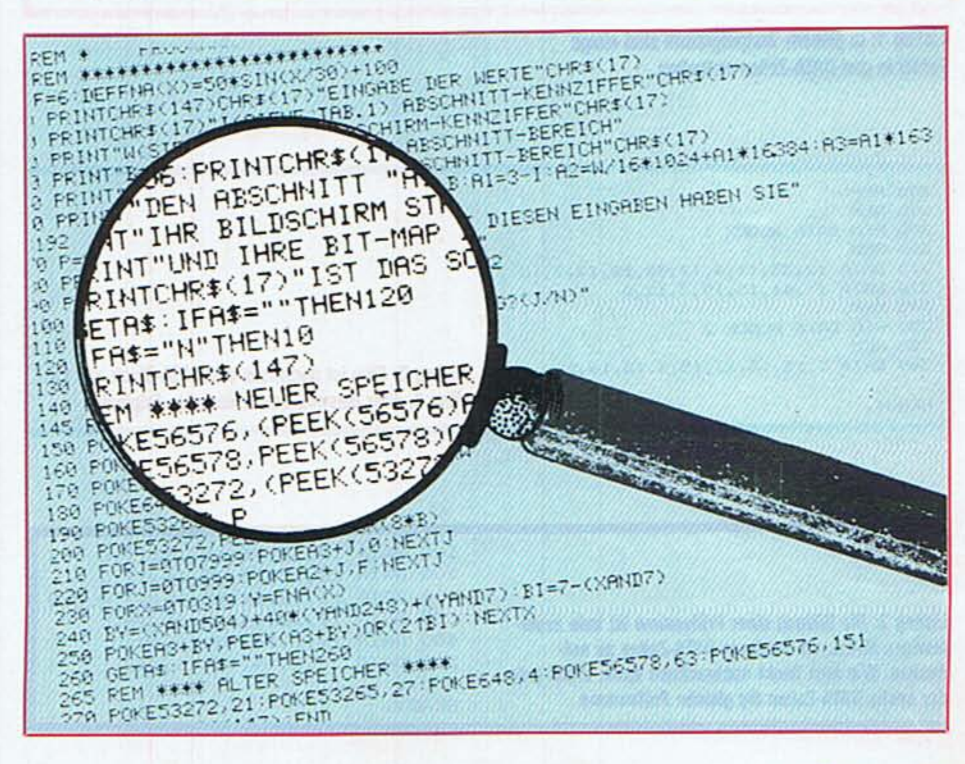
Wir erhalten wieder eine leicht zu überprüfende Liste aller DATA-Werte, diesmal aus Block 2. Bei I = 4 fällt uns sofort etwas auf. Am Bildschirm erscheint nämlich

```
I = 4  X = 15.4
```

Das ist die einzige Zahl mit Nachkommastellen, was bei dieser Art der Bildschirmausgabe sofort ins Auge sticht. Wir vergleichen den Wert mit der Eintragung im Originallisting und sehen sofort den Fehler: Wir haben beim Abtippen irrtümlich einen Punkt statt eines Kommas eingegeben. Die Korrektur ist leicht ausgeführt.

Danach löschen wir Zeile 225 mit dem GOSUB 1000 wieder und überzeugen uns durch einen abschließenden Probelauf vom einwandfreien Funktionieren des Programms.

Natürlich kann man nicht erwarten, daß sich alle Fehler so reibungslos lokalisieren lassen wie in unserem kleinen Beispiel. Gerade bei Fehlern in DATA-Zeilen kann die Suche sich namentlich bei längeren DATA-Blöcken um einiges schwieriger gestalten. Aber bei Programmen mit vielen DATA-Zeilen sind die hier beschriebenen Methoden zum Auffinden von versteckten Fehlern einfach unentbehrlich, wenn man einigermaßen schnell und sicher zum Ziel gelangen will. (ev)



```
1 REM DATA-TEST
2 REM =========
3 REM
4 REM
5 B=1024:REM ANFANGSADRESSE BILDSCHIRM BEI C 64
6 REM
7 PRINT"^S":REM BILDSCHIRM LOESCHEN
8 PRINT:PRINT:PRINT:PRINT
9 REM
100 REM DATA-BLOCK 1 LESEN
105 REM
110 FOR I=1 TO 17
120 READ X
130 S=S+X
140 NEXT I
150 PRINT "S =";S
160 IF S<>282 THEN PRINT "PRUEFSUMMENFEHLER":END
170 PRINT "OK"
190 REM
195 REM
200 REM DATA-BLOCK 2 LESEN
205 REM
210 FOR I=0 TO 8
220 READ X
230 POKE B+I,X
240 NEXT I
250 END
290 REM
295 REM
300 REM DATA BLOCK 1
305 REM
310 DATA 12,33,11,4,17,38,22,19,7,18,
320 DATA 11,41,15,19,3,12,0
345 REM
350 REM DATA BLOCK 2
355 REM
360 DATA 3,1513.13,15.4,15,18,5

READY.
```

Listing 1: In diesem Testprogramm sind einige Fehler in den DATA-Zeilen enthalten.

```
290 REM
295 REM
300 REM DATA BLOCK 1
305 REM
310 DATA 12,33,11,4,17,38,22,19.7,18
320 DATA 11,41,15,19,3,12,0
345 REM
350 REM DATA BLOCK 2
355 REM
360 DATA 3,15,13,13,15,4,15,18,5

READY.
```

Listing 2: Dies ist nochmals der DATA-Block aus Listing 1, aber diesmal das fehlerfreie »Original«.

```
100 DATA 12,13,14,15,16,17,0
200 DATA 13,12,14,15,16,17,0
300 DATA 13,12,14,15,16,17,0,0,0
320 DATA 11,13,14,15,16,18,0
400 DATA 23,2,14,15,16,17,0
500 DATA 23,2,14,15,16,,17

READY.
```

Listing 3: Die Bildung einer Prüfsumme ist kein zuverlässiges Mittel, um Fehler in DATA-Zeilen zu entdecken. Wie man leicht nachrechnen kann, ergibt jede der sechs DATA-Zeilen die gleiche Prüfsumme.

Der Volkscomputer und der große Bruder:

Adressenvergleich VC 20 — C 64

Der Commodore 64 wird oft als der »große Bruder« des VC 20 bezeichnet. Tatsächlich sind zum Beispiel Betriebssystem, Basicinterpreter und Schnittstellen weitgehend identisch. Daher können Basicprogramme, in denen keine Befehle und Funktionen wie POKE, PEEK, SYS und USR vorkommen, praktisch unverändert übertragen werden.

Leider wird man derartige Programme nur sehr selten finden, da beide Computer Grundlegendes gemeinsam haben: Die Hardwareeigenschaften werden von der Software — sprich vom Basic — kaum unterstützt. Immer dann, wenn die grafischen oder musikalischen Fähigkeiten von C 64 und VC 20 angesprochen werden, geschieht dies durch POKE-Befehle, auch wenn es nur darum geht, zum Beispiel die Farbe des Bildschirmrahmens einzustellen.

Das allein wäre ja noch nicht so schlimm. Nun ist aber die entsprechende Hardware, also Video- und Soundchip, bei beiden Computern grundlegend verschieden aufgebaut und belegt zudem noch völlig unterschiedliche Adressen im Speicher. So ist es kein Wunder, wenn das Umschreiben von C 64-Programmen auf den VC 20 (und umgekehrt) in der Regel mit erheblichen Schwierigkeiten verbunden ist. Wir wollen im folgenden versuchen, eine ganze Reihe dieser Schwierigkeiten aus dem Weg zu räumen. Als erste grobe Übersicht soll dabei die Tabelle 1 dienen. Auf weitere Einzelheiten wird im folgenden näher eingegangen.

	C 64	VC20 (Grundversion)	VC 20 (ab 8 K-Erw.)	VC 20
Video RAM	1024	7680	4096	Video-RAM
freier RAM-Bereich	2048	4096	4608	freier RAM-Bereich
Zeichengenerator-ROM	53248	32768	32768	Zeichengenerator-ROM
Video-Chip (VIC)	53248	36864	36864	Video & Sound Chip
Syntheziser Chip (SID)	54272	—	—	
I/O 1 (CIA #1)	56320	37136	37136	I/O 1 (VIA #1)
I/O 2 (CIA #2)	56576	37152	37152	I/O 2 (VIA #2)
FARB-RAM	55296	38400	37888	Farb-RAM
Basic-ROM	40960	49152	49152	Basic-ROM

Tabelle 1. Die Basisadressen wichtiger Speicher- und Registerbereiche

Bevor man daran gehen kann, für den jeweils anderen Computer geschriebene Software an den eigenen C 64 oder VC 20 anzupassen, müssen die Programme erst einmal in den Speicher gebracht werden. Ein Blick auf die Modulschächte beider Computer verrät sofort, daß es mit dem Austausch von Steckmodulen nicht allzu weit her sein kann: Das Modulformat ist völlig unterschiedlich.

Leider ist auch der Softwareaustausch per Programmkassette nur über einen Umweg zu realisieren. Zwar ist das Aufzeichnungsformat auf Cassette bei beiden Computern gleich, dennoch kann der eine Computer die Aufzeichnungen des anderen in den meisten Fällen nicht lesen. Der Grund hierfür liegt darin, daß der VC 20 eine höhere Taktfrequenz als der C 64 hat (VC 20-Programme sind um einiges schneller als entsprechende C 64-Programme). Dadurch gerät beim Lesen einer fremden Kassette die Synchronisation völlig aus den Fugen. Den einzigen Ausweg in dieser Situation bietet die Verwendung eines »großen« CBM der Serien 30xx oder 40xx als Vermittler. Die Taktfrequenz dieser CBM-Rechner liegt zwischen der des C 64 und der des VC 20, wodurch es zum Beispiel möglich ist, VC 20 Programme zunächst mit dem CBM zu laden, dann wieder abzuspeichern und nun wiederum mit dem C 64 zu laden.

Beim Programmaustausch per Floppy-Disk treten solche Probleme nicht auf. Hier kann man nach Herzenslust VC 20-Software in den C 64 laden oder auch umgekehrt. Hat man allerdings keine Floppy oder liegt das interessierende Programm nicht auf Diskette vor, dann bleibt in der Regel nur noch eins zu tun: Die Zähne zusammenbeißen und das Programm vom Listing abtippen. Dabei kann man dann auch gleich alle nötigen Programmänderungen vornehmen.

Leider gibt es viele Programme sowohl für den VC 20 als auch für den C 64, die man nicht durch Ändern einiger POKE-Adressen und kleinen Korrekturen am Bildschirmlayout an den jeweils anderen Computer anpassen kann.

Dazu gehören generell alle Programme, die hochauflösende Grafik verwenden. Die Prinzipien, nach dem die hochauflösende Grafik auf den beiden Computern realisiert ist, sind völlig unterschiedlich. Außerdem ist es natürlich von vorne herein völlig aussichtslos, ein C 64-Programm, das Sprites und Synthesizereffekte einsetzt, für den VC 20 umschreiben zu wollen. Umgekehrt gibt es eine solche Einschränkung allerdings nicht — der VC 20 kennt keine Sprites und sein Tongenerator läßt sich mit dem Synthesizer des C 64 allemal simulieren.

Sehr viel Vorsicht ist geboten, wenn das Programm längere Abschnitte in Maschinensprache enthält. Oftmals müssen diese Maschinenspracheroutinen in andere Spei-

cherbereiche verschoben werden, um gemeinsam mit dem Basic-Programm laufen zu können. Wir wollen uns an dieser Stelle aber nur mit den Anpassungen bei Basic-Programmen beschäftigen.

POKE-Befehle für Farbe und Bildschirm

Selbst die einfachsten Programme enthalten in der Regel Befehle, um Rahmen- und Hintergrundfarbe des Bildschirms einzustellen. Beim VC 20 werden beide Einstellungen gleichzeitig mit einem einzigen POKE-Befehl in Register 36879 durchgeführt (Tabelle 2). Der C 64 verwendet zwei getrennte Register, nämlich 53280 für die Rahmenfarbe und 53281 für die Hintergrundfarbe. Jeweils 16 Farben sind möglich (Tabelle 3). Zum Beispiel erzeugt der Befehl POKE 36879,95 beim VC 20 einen gelben Bildschirmrahmen und einen grünen Hintergrund. Mit POKE 53280,7 : POKE 53281,5 wird dasselbe am C 64 erreicht.

Beim C 64 enthält der Bildschirm 25 Zeilen zu je 40 Zeichen; der Bildschirmspeicher belegt die Adressen 1024 bis 2023. Der zugehörige Farbspeicher geht von 55296 bis 56295. Mit POKE 1024, 1 : POKE 55296, 2 erscheint zum Beispiel ein rotes »A« in der linken oberen Bildschirmecke. Der für den Anwender verfügbare RAM-Bereich beginnt bei Adresse 2048 (Bild 1).

Beim VC 20 wird die Angelegenheit etwas komplizierter. Die Anfangsadressen von Bildschirm- und Farbspeicher sind nämlich je nach Speicherausbau unterschiedlich (Bild 2). In der Grundversion und mit der 3-KByte-Erweiterung beginnt das Video-RAM bei Adresse 7680 und geht bis Adresse 8185. Das Farb-RAM belegt dann den Bereich von 38400 bis 38905. In dieser Konfiguration liegt der Bildschirmspeicher oberhalb des für den Anwender verfügbaren RAM-Bereiches, der ab Adresse 4096 (Grundversion) beziehungsweise 1024 (3-KByte-Erweiterung) beginnt. Sobald jedoch eine Speichererweiterung von mindestens 8 KBytes eingesteckt ist, wandert das Video-RAM nach »unten« und beginnt dann ab Adresse 4096. Dies geschieht, um für Basic-Programme einen zusammenhängenden Speicherbereich von der Adresse 4608 an aufwärts zu schaffen. Eine eventuell zusätzlich vorhandene 3-KByte-Erweiterung kann in diesem Fall nicht für Basic-Programme genutzt werden. Schließlich verändert auch noch das Farb-RAM seine Lage und startet jetzt bei Adresse 37888.

	BLK	WHT	RED	CYAN	PUR	GRN	BLU	YEL
Hintergrund	SCHW	WEISS	ROT	TÜRKIS	VIOLETT	GRÜN	BLAU	GELB
SCHWARZ	8	9	10	11	12	13	14	15
WEISS	24	25	26	27	28	29	30	31
ROT	40	41	42	43	44	45	46	47
TÜRKIS	56	57	58	59	60	61	62	63
VIOLETT	72	73	74	75	76	77	78	79
GRÜN	88	89	90	91	92	93	94	95
BLAU	104	105	106	107	108	109	110	111
GELB	120	121	122	123	124	125	126	127
ORANGE	136	137	138	139	140	141	142	143
HELLORAN.	152	153	154	155	156	157	158	159
ROSA	168	169	170	171	172	173	174	175
HELLTÜRKIS	184	185	186	187	188	189	190	191
HELLVIOL.	200	201	202	203	204	205	206	207
HELLGRÜN	216	217	218	219	220	221	222	223
HELLBLAU	232	233	234	235	236	237	238	239
HELLGELB	248	249	250	251	252	253	254	255

Tabelle 2. Farbkombinationen für Bildschirmrahmen und Hintergrund beim VC 20

VC 20 und C 64		nur C 64	
0	schwarz	8	orange
1	weiß	9	braun
2	rot	10	hellrot
3	türkis	11	grau 1
4	violett	12	grau 2
5	grün	13	hellgrün
6	blau	14	hellblau
7	gelb	15	grau 3

Tabelle 3. Farbwerte bei C 64 und VC 20

Der Bildschirm des VC 20 ist aufgeteilt in 23 Zeilen zu je 22 Zeichen. Insgesamt sind also 506 Bildschirmstellen vorhanden, das sind etwa halb soviele wie beim C 64. Eine Anpassung des Bildschirmlayouts ist also in fast allen Fällen erforderlich. In der Regel dürfte das kein Problem darstellen. Bei der Anpassung von VC 20-Programmen an den C 64 wird man des öfteren PRINT-Anweisungen zusammenfassen, da eine Bildschirmzeile beim C 64 fast doppelt soviele Zeichen wie eine entsprechende VC 20-Zeile aufnehmen kann. Im umgekehrten Fall ist das Einfügen von PRINT-Befehlen sinnvoll, um eine übersichtliche Bildschirmdarstellung zu erhalten. Ernste Schwierigkeiten kann es nur bei der Ausgabe von Tabellen auf dem Bildschirm geben. Eine sechsspaltige Zahlentabelle zum Beispiel läßt sich auf dem C 64 ganz gut darstellen, beim VC 20 wird man bei derartigen Versuchen unangenehm an die arg begrenzte Zeilenbreite erinnert. In solchen Fällen kann man versuchen, weniger interessante Spalten der Tabelle einfach fortzulassen. Wenn das nicht erwünscht oder möglich ist, hilft nur noch der CMD-Befehl, um die Ausgabe der entsprechenden Tabelle auf den Drucker umzuleiten.

Das Betriebssystem von C 64 und VC 20 ist weitgehend identisch. Natürlich gibt es einige Unterschiede bei den Ein-/Ausgabeoperationen, bedingt schon alleine durch das unterschiedliche Bildschirmformat. Fast alle nutzbaren Adressen in der Zeropage oder in den anderen vom Betriebssystem benutzten Speicherbereichen haben jedoch die gleiche Bedeutung. So befindet sich zum Beispiel bei beiden Computern der Kassettenpuffer zwischen den Adressen 828 und 1019 und mit »? PEEK(43) + 256 * PEEK(44)« erhält man beidesmal die Anfangsadresse des Basic-Programms.

Eine wichtige Ausnahme von dieser Regel ist der USR-Vektor. Beim VC 20 befindet er sich in den Speicherstellen 1 und 2 am Anfang der Zeropage, in Adresse 0 steht immer ein »JMP«-Befehl. Jedesmal bei der Ausführung der USR-Funktion ver-

zweigt das Basic zur Adresse 0 und aufgrund des dort stehenden »JMP« sofort weiter zur Adresse, die in sich in den Speicherzellen 1 und 2 befindet. Beim C 64 befinden sich dagegen am Anfang der Zeropage zwei Register des 6510-Mikroprozessors, so daß der USR-Vektor hier in die Adressen 785 und 786 verlegt wurde. Bei Programmen, welche die USR-Funktion verwenden, müssen diese unterschiedlichen Adressen unbedingt beachtet werden.

Wie aus den Bildern 1 und 2 hervorgeht, liegt der Basic-Interpreter beim C 64 in einem anderen Adreßbereich als beim VC 20. Da die Routinen aber bis auf Ausnahmen (USR-Funktion) völlig gleich sind, kann man die Adressen sehr einfach umrechnen: VC 20-Adressen zwischen $C000 und $DFFC entsprechen den C 64 Adressen zwischen $A000 und $BFFC, die Differenz ist also gerade $2000 oder dezimal 8192. In Adresse $BFFD steht beim C 64 ein Sprung nach $E000, um den RAM- und I/O-Bereich zwischen $C000 und $DFFF zu überbrücken. Durch diesen Sprungbefehl entsteht im folgenden eine Adressendifferenz um drei Bytes. Von allen C 64-Adressen zwischen $E000 und $E37A müssen daher diese drei Bytes abgezogen werden, um die entsprechenden VC 20-Adressen zu erhalten.

Einige häufig vorkommende POKE-Adressen wurden bisher noch nicht erwähnt. Gemeint sind die zur Programmierung von Musik und Geräuscheffekten benutzten Register. Leider ist die Art und Weise der Tonerzeugung bei beiden Computern völlig unterschiedlich, so daß sich keine äquivalenten POKE-Befehle angeben lassen.

Der C 64 verfügt nämlich über einen vollwertigen Synthesizer-Baustein, während der VC 20 nur »normale« Tongeneratoren besitzt. Tabelle 4 enthält für alle VC 20 Besitzer eine Übersicht über die beim C 64 zur Tonerzeugung benutzten Register. Diese Tabelle dient allerdings

Adresse			Inhalt
Stimme 1	Stimme 2	Stimme 3	
54272	54279	54286	Tonfrequenz (Low-Byte)
54273	54280	54287	Tonfrequenz (High-Byte)
54274	54281	54288	Tastaturverhältnis Rechteckgenerator (Low)
54275	54282	54289	Tastaturverhältnis Rechteckgenerator (High)
54276	54283	54290	Wellenform
54277	54284	54291	Anschlag/Abschwellen
54278	54285	54292	Halten/Ausklingen
	54296		Lautstärke

Mögliche Wellenformen: Dreieck (17), Sägezahn (33), Rechteck (65), Rauschen (129)

Tabelle 4. Die wichtigsten Register des C 64-Soundchips

NOTE	WERT	NOTE	WERT
C	135	G	215
C#	143	A^b	217
D	147	A	219
E^b	151	B	221
E	159	H	223
F	163	C	225
F#	167	C#	227
G	175	D	228
A^b	179	E^b	229
A	183	E	231
B	187	F	232
H	191	F#	233
C	195	G	235
C#	199	A^b	236
D	201	A	237
E^b	203	B	238
E	207	H	239
F	209	C	240
F#	212	C#	241

Stimmlagen-Befehle	X=	Funktion
POKE 36878,X	0 bis 15	setzt die Lautstärke
POKE 36874,X	128 bis 255	spielt Note
POKE 36875,X	128 bis 255	spielt Note
POKE 36876,X	128 bis 255	spielt Note
POKE 36877,X	128 bis 255	Geräuscheffekte

Tabelle 5. Notenwerte und Tongeneratoradressen beim VC 20

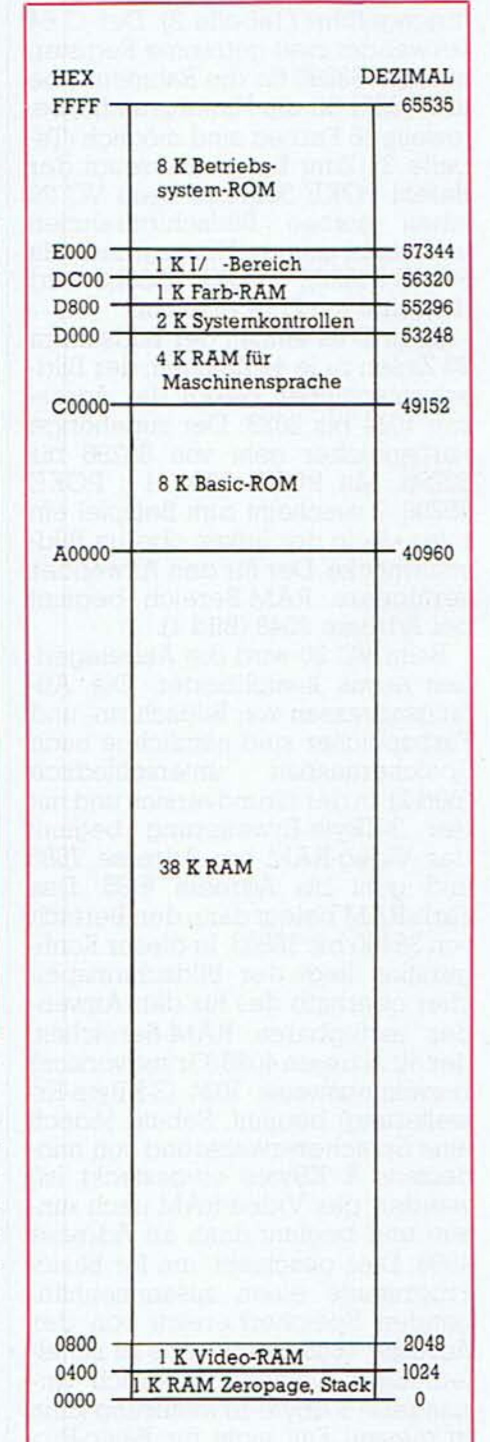


Bild 1. Speicherbelegung des Commodore 64 im Grundzustand. Im Bereich $D000 bis $DFFF sind zusätzlich 4 KByte Zeichengenerator-ROM überlagert

wirklich nur zur Orientierung, welche POKE-Befehle beim C 64 zur Tonerzeugung dienen. Eine Simulation des C 64-Synthesizers ist mit dem VC 20 nicht möglich. Es empfiehlt sich daher, bei der Programmanpassung zunächst einmal alle derartigen POKE-Befehle fortzulassen und später eigene Sound-Routinen einzufügen.

C 64-Besitzer haben es hier etwas besser: Mit etwas Geschick und einem guten Handbuch zur Musikprogrammierung können sie dem C 64 auch alle VC 20-Töne entlocken. Als Referenz hierzu kann Tabelle 5 mit den Daten zu den Tongeneratoren des VC 20 dienen. Im Zweifelsfalle sollte man aber auch hier eher auf originalgetreue Tonuntermalung verzichten und sich damit eine ganze Menge Arbeit sparen.

Und noch ein Punkt, wo der VC 20 hardwäremäßig benachteiligt ist: Der C 64 verfügt nämlich gleich über zwei Joystickports (mit den Adressen 56320 und 56321), während der VC 20 sich mit einem Anschluß zufrieden geben muß, der zu allem Überfluß auch noch ein recht kompliziertes Abfrageprogramm erfordert. Der Feuerknopf und die Schalter 0, 1 und 2 des Joysticks werden nämlich beim VC 20 über VIA #1 gelesen, während der Zustand von Schalter 3 über VIA #2 abgefragt wird. Normalerweise sind Joystickabfragen in Programmen leicht zu finden. Halten Sie beim VC 20 nach PEEKs in die Speicherstellen 37137 und 37152 Ausschau und beim C 64 nach entsprechenden Abfragen der Adressen 56320 und 56321. Anschließend muß die gesamte Joystickroutine für den jeweiligen Rechner neu geschrieben werden, da die Art der Abfrage einfach zu unterschiedlich ist.

Bei der Steuerung mittels Drehreglern (sogenannten Paddles) ist die Anpassung wesentlich einfacher zu realisieren. Die Paddle-Werte werden beim C 64 aus den Registern 54297 und 54298 ausgelesen. Beim VC 20 sind es die Register 36872 und 36873. Das Umschreiben besteht hier also lediglich im Einsetzen der entsprechenden Adressen in die PEEK-Befehle.

Wo sich das Umschreiben lohnt

An dieser Stelle muß noch einmal deutlich darauf hingewiesen werden, daß die Zahl derjenigen Programme, die mit vertretbarem Aufwand vom C 64 zum VC 20 oder umgekehrt übertragen werden können, doch verhältnismäßig gering ist. Bei Spielprogrammen ist in der Regel äußerste Vorsicht geboten, da hier aus Geschwindigkeitsgründen zumeist mit Routinen in Maschinensprache gearbeitet wird. Außerdem ist bei Spielen in der Regel das Bildschirmlayout fest vorgegeben, so daß wegen der unterschiedlichen Bildschirmkapazität beider Computer sehr wahrscheinlich größere Probleme auftreten werden. Von Sprites und hochauflösender Grafik einmal ganz zu schweigen.

Dagegen gibt es viele Anwendungsprogramme, welche den Computer nicht zu einer hochspezialisierten Spielmaschine machen, sondern ganz einfach Problemlösungen in Basic anbieten. Dabei kann es sich beispielsweise um ein Textverarbeitungsprogramm, eine Lagerverwaltung oder ganz einfach um ein Programm zum Ausdrucken eines Jahreskalenders handeln. Für fast alle derartigen Programme sollte es möglich sein, eine Anpassung mit Hilfe der hier abgedruckten Tabellen und Adressenvergleiche vorzunehmen. (ev)

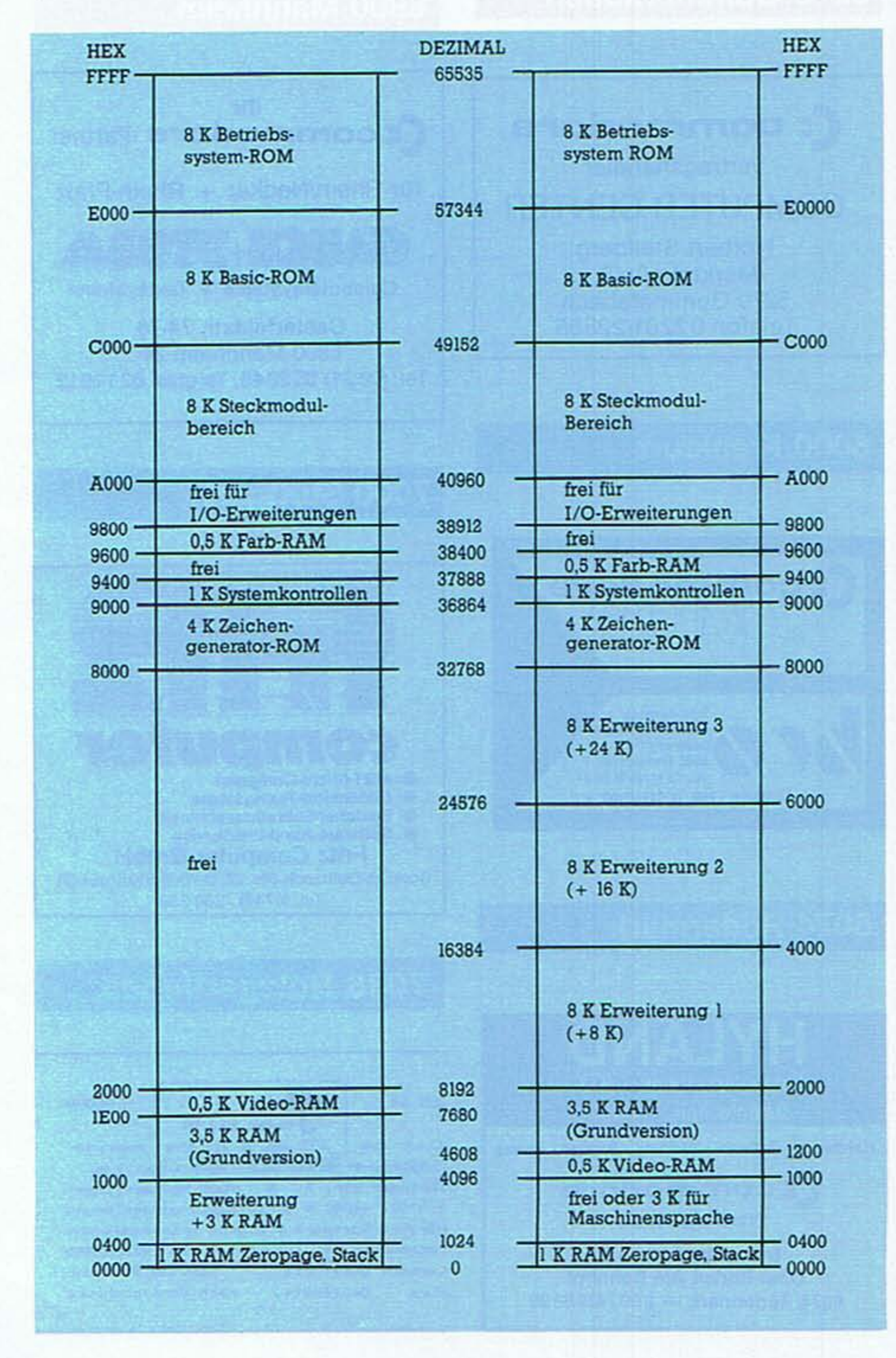


Bild 2. Speicherbelegung des VC 20 bis 3-KByte-Erweiterung (links) und ab 8-KByte-Erweiterung (rechts)

DATEN IM (RELATIVEN) DIREKTZUGRIFF

In diesem Bericht werden die Unterschiede zwischen der relativen und der sequentiellen Datei aufgezeigt und anhand eines Beispielprogramms die Programmierung einer relativen Datei erklärt.

Kommen wir gleich zu den Vor- und Nachteilen. Der Hauptvorteil der relativen Datei ist der schnelle Zugriff auf Daten. Ein weiterer Vorteil ist, daß eigentlich nur wenig Speicherplatz des Computers notwendig ist, nämlich nur so viel, wie das Programm selbst benötigt. Deshalb ist eine sinnvolle Anwendung dieser Dateiform auch mit dem VC 20 in der Grundversion möglich. Allerdings ist ein Diskettenlaufwerk unbedingt notwendig. Die sequentielle Datei hingegen benötigt viel Speicherplatz im Computer, und ein schneller Zugriff auf Daten ist nicht einfach zu realisieren, zumindest muß die gesamte Datei erst in den Speicher des Computers geladen werden, bevor man sinnvoll mit ihr arbeiten kann. Jedoch ist nicht unbedingt ein Diskettenlaufwerk notwendig, die Datasette tut's auch, wenn auch erheblich langsamer. Der Nachteil der relativen Datei ist die Art des Zugriffs auf bestimmte Daten. Er ist nämlich nur über die Datensatznummer möglich. Das heißt: Angenommen, wir haben uns eine Adressendatei aufgebaut und auch schon eine Anzahl Adressen eingegeben. Wenn wir uns jetzt die Adresse von Anton Huber ausgeben lassen wollen, so ist das nicht möglich, indem wir den Namen »Huber« eingeben und dann das Ergebnis, seine Adresse, auf einen Schlag vor uns haben. Dazu müßten wir seine Satznummer kennen. Vereinfacht kann man sagen, daß die Satznummer angibt, die wievielte Adresse gemeint ist. Satznummer 14 bedeutet also die 14. Adresse. Diese Adresse findet der Computer, weil er vom Dateianfang ausgeht, den er sich merkt, und dann 13 Datensätze überspringt (auf den nächsten Datensatz positioniert), um direkt den 14. Datensatz zu lesen (daher der Name relative Datei: relativ zum Dateianfang).

```
100 REM ******************************
110 REM * ADRESSENDATEI 64'ER/7 REL  *
120 REM ******************************
130 CLOSE 15:OPEN 15,8,15
140 CLOSE 1:OPEN 1,8,2,"ADR.REL,L,"+CHR$
(82)
1000 REM ------------------------------
1010 REM -    AUSWAHL                 -
1020 REM ------------------------------
1030 :
1040 PRINT "L      RELATIVE DATEI 64'ER 7"
1050 PRINT :PRINT :PRINT
1060 PRINT "  1 = DATEI EINRICHTEN"
1070 PRINT "  2 = DATEN EINGEBEN"
1080 PRINT "  3 = SUCHEN UND ANZEIGEN"
1090 PRINT
1100 PRINT "  X = PROGRAMMENDE"
1110 PRINT :PRINT
1120 PRINT "WAEHLE"
1130 GET R$:IF R$="" THEN 1130
1140 IF R$="1" THEN GOSUB 11000
1150 IF R$="2" THEN GOSUB 4000
1160 IF R$="3" THEN GOSUB 3000
1170 IF R$="X" THEN CLOSE 1:CLOSE 15:END

1180 GOTO 1000
2000 REM --------------------------------
2010 REM -   BILDSCHIRMANZEIGE
2020 REM --------------------------------
2030 :
2040 PRINT "L      ANZEIGE DATENSATZ"
2050 PRINT
2060 PRINT " NUMMER ";RN:PRINT
2070 PRINT
2080 PRINT " NAME    : "NV$" "NN$
2090 PRINT " STRASSE : "SR$
2100 PRINT " PLZ/ORT : "PL$" "OT$
2110 PRINT " TELEFON : "TE$
2120 PRINT :PRINT :PRINT
2130 RETURN
2140 :
```

Mit diesem Programm können Sie bis zu 1978 Datensätze (Adressen) verwalten. Das gilt für C 64 und für den VC 20!

```
3000 REM ---------------------------------
3010 REM - SUCHEN
3020 REM ---------------------------------
3030 :
3040 GOSUB 13000:REM  DATENSATZNR.EINGAB
E
3050 GOSUB 9000:REM  LESEN
3060 IF RC$<>CHR$(255) THEN 3120
3070 PRINT "♥"
3080 PRINT :PRINT :PRINT
3090 PRINT " NOCHT NICHT BESCHRIEBEN"
3100 PRINT :PRINT
3110 GOTO 3140
3120 GOSUB 5000:REM  AUFTEILEN
3130 GOSUB 2000:REM  ANZEIGEN
3140 PRINT " DRUECKE TASTE"
3150 GET R$:IF R$="" THEN 3150
3160 R$=""
3170 RETURN
3180 :
4000 REM ---------------------------------
4010 REM -         EINGABE              -
4020 REM ---------------------------------
4030 :
4040 GOSUB 13000:REM  DATENSATZNR.
4050 GOSUB 6000:REM  EINGABE
4060 GOSUB 7000:REM  VERKETTEN
4070 GOSUB 8000:REM  SPEICHERN
4080 RETURN
4090 :
5000 REM ---------------------------------
5010 REM  AUFTEILEN DATENSATZ IN FELDER
5020 REM ---------------------------------
5030 :
5040 NV$=LEFT$(RC$,15)
5050 NN$=MID$(RC$,16,15)
5060 SR$=MID$(RC$,31,20)
5070 PL$=MID$(RC$,51,4)
5080 OT$=MID$(RC$,55,15)
5090 TE$=MID$(RC$,70,12)
5100 RETURN
6000 REM ---------------------------------
6010 REM - EINGABE NEUE DATEN           -
6020 REM ---------------------------------
6030 :
6040 PRINT "♥"
6050 PRINT "     EINGABE NEUE DATEN"
6060 PRINT :PRINT
6070 PRINT "     NUMMER ";KN
6080 PRINT :PRINT
6090 INPUT "NACHNAME ";NN$:NN$=LEFT$(NN$
.15)
6100 INPUT "VORNAME  ";NV$:NV$=LEFT$(NV$
,15)
6110 INPUT "STRASSE  ";SR$:SR$=LEFT$(SR$
,20)
6120 INPUT "POSTLZ.  ";PL$:PL$=LEFT$(PL$
,4)
6130 INPUT "ORT      ";OT$:OT$=LEFT$(OT$
,15)
6140 INPUT "TELEFON  ";TE$:TE$=LEFT$(TE$
,12)
6150 PRINT :PRINT
6160 PRINT " ADRESSE OK (J/N) ?"
6170 GET R$:IF R$="" THEN 6170
6180 IF R$="N" THEN 6040
6190 IF R$<>"J" THEN 6170
6200 RETURN
```

Programmierung einer relativen Datei (Fortsetzung)

Dazu ist eine Voraussetzung notwendig: Die Datensätze müssen alle die gleiche Länge haben. Wie das realisiert wird, erkläre ich noch.

Um einer möglichen Frustration vorzubeugen, sei gesagt, daß auch eine direkte Suche über einen Namen möglich ist, indem man die Vorteile der relativen Datei mit der der sequentiellen Datei verknüpft.

Eine relative Datei besteht im Prinzip aus 3 Teilen:

1. Einrichten einer Datei
2. Speichern (Schreiben) eines Datensatzes
3. Lesen eines Datensatzes

Gehen wir diese Teile Schritt für Schritt durch. Anhand des Listings können Sie diese Schritte mitverfolgen.

1. Einrichten einer relativen Datei

Um mit einer relativen Datei arbeiten zu können, müssen 2 Kanäle zur Floppy geöffnet werden. Zum ersten ist das der Kommandokanal (Kanal 15) und zweitens ein beliebiger anderer Kanal. Der Kommandokanal ist notwendig um den Positionierbefehl zu übertragen. Mit dem anderen Kanal wird die Datei eröffnet und bearbeitet.

1.a) Öffnen des Kommandokanals
130 CLOSE 15:OPEN 15,8,15

Sicherheitshalber sollte man vor jedem OPEN ein CLOSE setzen, um eine eventuelle Fehlermeldung »FILE OPEN ERROR« zu verhindern.

1.b) Eröffnen der Datei und Festsetzen der Datensatzlänge

Wie schon erwähnt, ist ein Merkmal der relativen Datei, daß jeder Datensatz die gleiche Länge besitzen muß. Diese Angabe muß beim Einrichten der Datei angegeben werden. Das geschieht mit folgendem Befehl:

OPEN lfn,ga,kanal,"dateiname,L," + CHR$(datensatzlänge)

(im Listing Zeile 140 und 11120)

lfn = logische filenummer

ga = Geräteadresse (normalerweise 8)

kanal (2-14)

dateiname = Name der Datei,wird so im Directory abgelegt, im Beispiel »ADR.REL«.

L = Kennzeichen für eine relative Datei

datensatzlänge = Summe aller Feldlängen (1-254)

Der Buchstabe »L« sagt dem DOS, daß eine relative Datei eröffnet wird. Diesem Buchstaben muß die Angabe der Datensatzlänge folgen. Sie wird mit dem CHR$ gesandt. Im Beispiel setzt sich ein Datensatz folgendermaßen zusammen:

NAME = 15 Buchstaben
VORNAME = 15 Buchstaben
STRASSE = 20 Buchstaben
POSTLZ = 4 Buchstaben
ORT = 15 Buchstaben
TELEFON = 12 Buchstaben

Das ergibt insgesamt 81 Buchstaben pro Datensatz. Da jeder Datensatz (auch RECORD genannt) mit einem RETURN abgeschlossen wird, erhöht sich die Datensatzlänge auf insgesamt 82 Zeichen.

1.c) Positionieren

Der Computer findet einen bestimmten Datensatz über die Datensatznummer, indem er sich den Dateianfang merkt und die entsprechende Anzahl Datensätze überspringt. Genauer gesagt, er positioniert auf diesen Datensatz. Und das wird mit dem Positionierbefehl erledigt. Dieses Kommando wird über den Kommandokanal (15) der Diskette gesandt. Seine Form ist:

PRINT # lfn,"P" + CHR$(kanal+ CHR$(low) + CHR$(high) + CHR$(byte)

(siehe dazu Zeile 11130 - 11190)

Die Parameter »low« und »high« geben die Datensatz(=Record)nummer an. Da ein Byte maximal den Wert 255 annehmen kann, die Recordnummer aber höher sein darf, muß sie aufgeteilt werden in ein Low-Byte und ein High-Byte. Das geschieht einfach mit den Anweisungen

HB=INT(RN/256)
LB=RN-HB*256

RN = Recordnummer
HB = Höherwertiges Byte
LB = Niederwertiges Byte

Der letzte Parameter (Byte) positioniert auf eine bestimmte Stelle innerhalb eines Records. Beispiel:
PRINT # 15,"P" + CHR$(2) + CHR$(12) + CHR$(1) + CHR$(8)

Hier wird auf das 8. Byte des 268. Records positioniert. Die 268 ist die Recordnummer und wird aufgeteilt in ein Low-Byte(12) und ein High-Byte(1). (HighByte*256 + Low-Byte=Recordnummer). Beim Schreiben eines Records muß jedoch immer auf das erste Byte positioniert werden.

Zum Schluß wird die Datei freigegeben.

1.d) Freigeben der Datei
PRINT # 15,CHR$(255)

Diese Anweisung bewirkt, daß alle Records, die unter der angegebenen Recordnummer liegen, mit CHR$(255) beschrieben werden, sofern sie noch nicht anders belegt wurden. Das dauert seine Zeit. Je mehr Datensätze eingerich-

```
6210 :
7000 REM ---------------------------
7010 REM     VERKETTEN DER FELDER   -
7020 REM ---------------------------
7030 :
7040 BL$="                    "
7050 RC$=NV$+LEFT$(BL$,15-LEN(NV$))
7060 RC$=RC$+NN$+LEFT$(BL$,15-LEN(NN$))
7070 RC$=RC$+SR$+LEFT$(BL$,20-LEN(SR$))
7080 RC$=RC$+PL$+LEFT$(BL$,4-LEN(PL$))
7090 RC$=RC$+OT$+LEFT$(BL$,15-LEN(OT$))
7100 RC$=RC$+TE$+LEFT$(BL$,12-LEN(TE$))
7110 RETURN
7120 :
8000 REM ---------------------------
8010 REM  - SPEICHERN DATEN AUF DISK -
8020 REM ---------------------------
8030 :
8040 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR$
(HB)+CHR$(1)
8050 PRINT# 1,RC$
8060 RETURN
8070 :
9000 REM ---------------------------
9010 REM  - LESEN DATENSATZ VON DISK -
9020 REM ---------------------------
9030 F=0
9040 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR$
(HB)+CHR$(1)
9050 INPUT# 15,ER:REM  FEHLERKANAL
9060 IF ER<>50 THEN 9110:REM  RECORD NOT
 PRESENT-ERROR
9070 PRINT "  DATENSATZNUMMER ZU HOCH"
9080 PRINT :PRINT " DRUECKEN SIE EINE TA
STE"
9090 GET R$:IF R$="" THEN 9090
9100 RUN
9110 INPUT# 1,RC$
9120 IF ASC(RC$)<>255 THEN 9140
9130 REM  DATENSATZ IST NOCH FREI
9140 RETURN
11000 REM ---------------------------
11010 REM  - NEUE DATEI ANLEGEN
11020 REM ---------------------------
11030 PRINT "  ACHTUNG, EINE BESTEHENDE
 DATEI MIT"
11040 PRINT :PRINT "   DEM NAMEN 'ADR.RE
L' WIRD GELOESCHT"
11050 PRINT
11060 PRINT "  (W)EITER   (Z)URUECK"
11070 GET R$:IF R$="" THEN 11070
11080 IF R$<>"W" THEN RETURN
11090 PRINT
11100 PRINT :PRINT "BITTE WARTEN"
11110 CLOSE 15:OPEN 15,8,15,"S:ADR.REL"
11120 CLOSE 1:OPEN 1,8,2,"ADR.REL,L,"+CH
R$(82)
11130 PRINT "WIEVIELE DATENSAETZE SOLL D
IE DATEI "
11140 PRINT "VERWALTEN? ";
11150 INPUT RN
11160 HB=INT(RN/256)
11170 LB=RN-HB*256
11180 PRINT :PRINT "BITTE WARTEN"
11190 PRINT# 15,"P"+CHR$(2)+CHR$(LB)+CHR
$(HB)+CHR$(1)
11200 INPUT# 15,ER:REM  FEHLERKANAL
11210 IF ER<>52 THEN 11250:REM  DATEI ZU
 GROSS
```

Programmierung einer relativen Datei (Fortsetzung)

```
11220 PRINT "      DATEI  ZU GROSS"
11230 PRINT :PRINT " DRUECKEN SIE EINE T
ASTE"
11240 GET R$:IF R$="" THEN 11270
11250 PRINT# 1,CHR$(255)
11260 CLOSE 1:CLOSE 15
11270 RUN
13000 REM ------------------------------
13010 REM - EINGABE DATENSATZNUMMER
13020 REM ------------------------------
13030 :
13040 PRINT "♥"
13050 PRINT :PRINT :PRINT
13060 PRINT "  EINGABE DER DATENSATZNUMM
ER"
13070 INPUT RN
13080 HB=INT(RN/256)
13090 LB=RN-HB*256
13100 RETURN

READY.
```

Listing zur relativen Datei (Schluß)

tet (freigegeben) werden, desto mehr Zeit beansprucht diese Arbeit. Aber erst diese Prozedur erlaubt ein fehlerfreies Lesen und schnelles Schreiben von Datensätzen. Wollen Sie einen Record beschreiben, der oberhalb des zuletzt freigegebenen Records liegt, so werden automatisch alle Records, die zwischen dem letzten und dem gerade beschriebenen Record liegen, freigegeben, das heißt, mit CHR$(255) beschrieben. Um diese Prozedur zu vermeiden, sollten Sie nur das erste Mal, beim Einrichten der Datei, die maximal zu erwartende Anzahl Records freigeben. Die Fehlermeldung »RECORD NOT PRESENT«, die dann im Floppy-Fehlerkanal erscheint, kann ignoriert werden, da dieser Record nur beschrieben (freigegeben) wird.

2. Speichern (Schreiben) eines Datensatzes

Zum Speichern eines Datensatzes sind folgende Funktionen durchzuführen:

2.a) Eingabe der Recordnummer und Aufteilung in Low- und High-Byte (siehe oben).

2.b) Eingabe der Daten (zum Beispiel über Input). Hier wird auch sichergestellt, daß die einzelnen Felder nicht länger werden, als vorher geplant wurde.

2.c) maximale Datensatzlänge bilden.

Das wird im Beispiel (siehe Listing) in Zeile 7000 bis 7999 durchgeführt. Es wird ein String mit der maximalen Länge (im Beispiel 81) gebildet. Dazu wird vorher eine Variable (im Listing BL$) definiert, mit der Anzahl Leerstellen, die das längste Feld unseres Datensatzes besitzt (im Beispiel das Feld »STRASSE« mit 20 Zeichen, also enthält BL$ 20 Leerstellen). Jedes Feld unseres Datensatzes wird entsprechend seiner vorher festgelegten Länge und abhängig von der wirklichen Länge bei der Eingabe mit den notwendigen Leerstellen aufgefüllt. Im Beispiel hat die Variable RC$ dann die Länge von 81 Zeichen und sie enthält den vollständigen Datensatz.

2.d) Speichern

Das Speichern dieses Datensatzes ist dann schnell geschehen: Es wird auf die vorher (in 2.a) angegebene Recordnummer positioniert und danach mit PRINT#1,RC$ gespeichert.

Mit PRINT# kann man jedoch nur Datensätze abspeichern, die nicht länger als 88 Zeichen sind. Sonst muß man die Daten Zeichen für Zeichen mittels einer GET# -Schleife schreiben und lesen.

3. Lesen eines Datensatzes

Hier erfolgt die Reihenfolge fast umgekehrt wie beim Schreiben. Zuerst muß jedoch auch hier die Nummer des gesuchten Datensatzes eingegeben werden. Nach entsprechender Aufteilung in Low- und High-Byte wird auf den Record positioniert und anschließend mit INPUT#1, RC$ von der Disk gelesen.

3.a) Eingabe Recordnummer und Aufteilung in Low- und High-Byte.

3.b) Lesen des Datensatzes

3.c) Aufteilen des Datensatzes

Dieser Vorgang muß umgekehrt dem Verketten (2.c) geschehen. Es wird ja lediglich der komplette Datensatz in die Variable RC$ gelesen. In Zeile 5000 bis 5999 werden wieder die einzelnen Felder gebildet. Danach fehlt nur noch die

3.d) Anzeige des Datensatzes

Das wars auch schon. Wenn Sie sich einmal alles gut durch den Kopf gehen lassen und die Problematik am Beispiel durchgehen und ausprobieren, können Sie in Zukunft Ihre eigene relative Datei programmieren.

Es ist gar nicht so schwierig, wie es auf den ersten Blick aussehen mag. Sie können auch Teile dieses Programms in Ihre eigenen einbauen. Schließlich ist das der Vorteil eines strukturierten Programmablaufs.

Zum Schluß möchte ich Sie noch auf einige Besonderheiten im Listing hinweisen.

Zeile 11110: Löschen einer eventuell bestehenden relativen Datei.

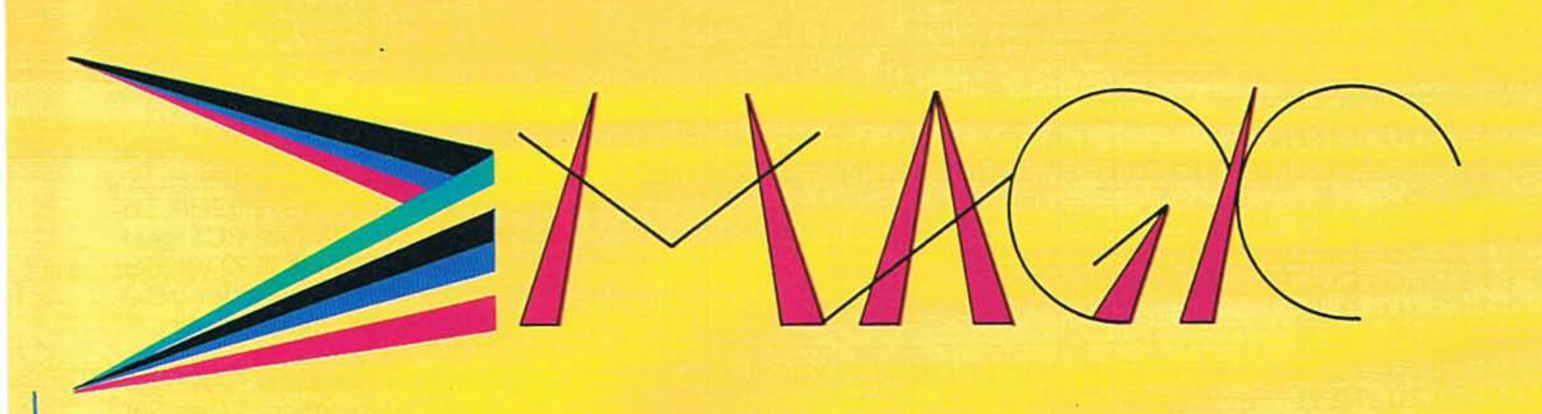
Zeile 11200-11220: Lesen des Fehlerkanals. Wenn der Fehler 52 gemeldet wird, bedeutet dies, daß die Anzahl Datensätze, die Sie angegeben haben, die Speicherkapazität der Diskette sprengen würde. Der Fehler 50 (RECORD NOT PRESENT, siehe auch Zeile 9050) bedeutet, daß Sie einen Datensatz lesen wollen, der nicht freigegeben wurde. Beim Schreiben braucht diese Fehlermeldung nicht beachtet werden (siehe 1.d).

Zeile 3060: Wenn ein gelesener Datensatz den Wert CHR$(255) besitzt, wird davon ausgegangen, daß er noch nicht beschrieben wurde. Deshalb wäre es auch sinnlos, Ihn anzeigen zu wollen. Dieser Vergleich wird in Zeile 9120-9130 nocheinmal in anderer Form durchgeführt, kann dort jedoch weggelassen werden.

Zeile 1170: Wenn das Programm beendet werden soll, muß der zur Positionierung notwendige Kanal 15 wieder geschlossen werden.

Anzumerken ist noch, daß nicht mehr als eine relative Datei gleichzeitig geöffnet werden kann. Es ist lediglich möglich, noch eine zusätzliche sequentielle Datei zur gleichen Zeit zu nutzen. Und diese Möglichkeit erlaubt uns, auch komfortablere Suchkriterien als über die Recordnummer einzusetzen. Doch darüber mehr in der übernächsten Ausgabe des 64'er Magazins.

(gk)



Bildschirm statt Schreibtisch



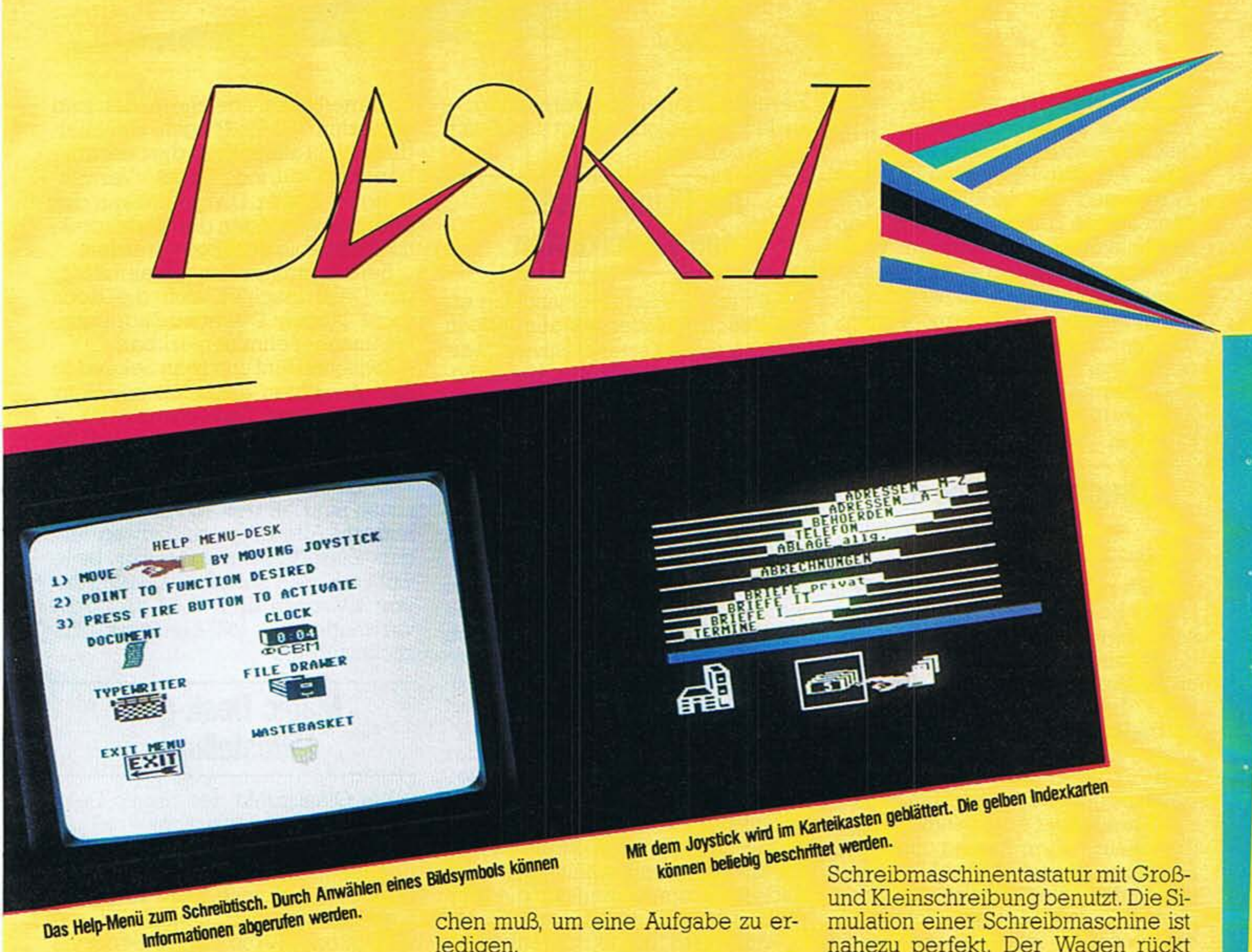
So erscheint der »magische Schreibtisch« auf dem Bildschirm

Die Schreibmaschine wird sehr realistisch simuliert

Magic Desk, der »magische Schreibtisch«, ist ein völlig neuartig konzipiertes Datenverwaltungsprogramm für den Commodore 64. Man kann es verwenden zum Schreiben, Ablegen, Vervielfältigen von Schriftstücken jeder Art, zum Führen von Adreßkarteien, Inhaltsverzeichnissen oder einfach als elektronischen Notizblock — und das alles, ohne von Arbeitsweise und Programmierung eines Computers etwas verstehen zu müssen.

Es ist völlig ausreichend, wenn man mit Schreibmaschine und Karteikasten umgehen kann und außerdem weiß, wozu ein Papierkorb gut ist. Denn eben diese Gegenstände werden auf dem Bildschirm mittels Farbgrafik in natürlicher Weise um einen Schreibtisch gruppiert dargestellt und können mit einem Joystick in einfacher Weise angewählt und aktiviert werden.

Um mit Magic Desk, das als Steckmodul geliefert wird, arbeiten zu können, benötigt man neben einem Commodore 64 (oder Executive 64) in jedem Falle ein Floppy-Laufwerk sowie Joystick und Drucker. Modul einstecken, Joystick an Control Port 2 anschließen und eine leere Diskette ins Laufwerk einlegen — und schon kann's losgehen. Der Bildschirm zeigt zunächst ziemlich formatfüllend einen Schreibtisch, daneben aufeinandergetürmt drei große Karteikästen (Bild 1). Auf der Schreibtischplatte befinden sich ein Telefon, ein Taschenrechner, eine Schreibmaschine, ein Finanzjournal und ein Satz Karteikarten. Ein Pa-



Das Help-Menü zum Schreibtisch. Durch Anwählen eines Bildsymbols können Informationen abgerufen werden.

Mit dem Joystick wird im Karteikasten geblättert. Die gelben Indexkarten können beliebig beschriftet werden.

pierkorb und eine Digitaluhr sind ebenso vorhanden. Von diesen Gegenständen kann man mit Magic Desk I jedoch nur Schreibmaschine, Papierkorb, Digitaluhr sowie die drei Karteikästen benutzen. Die anderen Utensilien sind für spätere Magic Desk — Versionen vorgesehen.

Ebenfalls auf dem Bildschirm erscheint eine Hand mit ausgestrecktem Zeigefinger. Diese Hand kann mit dem Joystick beliebig über den Bildschirm bewegt werden. Um eine Funktion zu aktivieren, positioniert man die Hand so, daß der ausgestreckte Zeigefinger auf das entsprechende Gerät deutet und drückt den Feuerknopf am Joystick. Das ausgewählte Gerät erscheint daraufhin weiß eingerahmt und das Schirmbild wechselt in den meisten Fällen. Nun kann man weitere Unterfunktionen aufrufen, indem man den Handzeiger auf andere grafisch dargestellte Gegenstände deuten läßt oder die Tastatur benutzt. Das Arbeiten gestaltet sich so sehr komfortabel und übersichtlich, ohne daß man sich darum zu kümmern hat, welche Schritte der Computer machen muß, um eine Aufgabe zu erledigen.

Briefeschreiben leichtgemacht

Nehmen wir zum Beispiel das Briefeschreiben mit anschließender Archivierung einer Kopie. Mit Magic Desk geht das mindestens genauso problemlos wie an einem realen Schreibtisch. Mittels Joystick läßt man den Handzeiger auf die auf dem Bildschirm abgebildete Schreibmaschine fahren und drückt den Feuerknopf. Sofort wechselt das Schirmbild und zeigt im unteren Drittel die grafischen Darstellungen von fünf Objekten: Schreibtisch, Tabulatoren (der Schreibmaschine), die Schreibmaschine selbst, Drucker und Papierkorb.

Die Schreibmaschine ist weiß umrahmt und der bekannte Handzeiger deutet darauf. Solange der weiße Rahmen um die Schreibmaschine existiert, läßt sich der Handzeiger nicht bewegen. Dafür kann man nun die Schreibmaschine benutzen, deren Wagen ungefähr in der Bildschirmmitte sehr realistisch dargestellt ist (Bild 2). Die Computertastatur wird nun als normale Schreibmaschinentastatur mit Groß- und Kleinschreibung benutzt. Die Simulation einer Schreibmaschine ist nahezu perfekt. Der Wagen rückt mit jedem Anschlag eine Stelle nach links, wobei maximal 80 Zeichen in einer Zeile möglich sind, von denen aber immer nur 40 Zeichen im Bild sind (horizontales Scrolling).

Einige Stellen vor Erreichen des Zeilenendes ertönt eine Warnklingel, bei Betätigen der RETURN-Taste wird ein Wagenrücklauf mittels Grafik und entsprechendem Geräusch simuliert. Mit den Funktionstasten F5 und F7 können Tabulatoren gesetzt oder gelöscht werden, mit F3 erfolgt ein schneller Wagenvorlauf auf die nächste Tabulatorposition. Mittels Joystick oder Cursortasten kann man sich auf dem dargestellten Papierbogen in alle vier Richtungen bewegen und mit der DEL-Taste beliebige Textbereiche löschen.

Vorsicht ist geboten beim Umgang mit der DEL- und der SPACE-Taste. DEL löscht das Zeichen, auf das die Maschine gerade positioniert ist und hinterläßt einen Freiraum, zieht den Text also anders als beim normalen Basic-Betrieb nicht zusammen. Die SPACE-Taste wirkt dagegen genauso wie ein »Cursor rechts«, verändert also keine Zeichen. Das ist beim Überschreiben von Textstellen des

öfteren sehr störend, da unerwünschte Zeichen explizit mit DEL gelöscht werden müssen.

Bis auf das Überschreiben von Texten und das Löschen von Zeichen gibt es keine weiteren Editiermöglichkeiten, insbesondere ist es unmöglich, zusätzlichen Text einzufügen oder Textstellen zu kürzen. Diese fehlenden Editiermöglichkeiten sind sicherlich ein Schwachpunkt des Programms, auch wenn man berücksichtigt, daß Magic Desk kein hochspezialisierter Texteditor ist und auch keiner sein soll.

Druckerausgabe auf Knopfdruck

Doch nun weiter in unserem Beispiel ! Hat man den Text fertig geschrieben oder ist man am Ende einer Seite angelangt (jede Seite hat 66 Zeilen), dann kann man den Text auf dem Drucker ausgeben und/oder in einem der drei Karteikästen archivieren. Das Ausdrucken einer Seite gestaltet sich dabei denkbar einfach: Man drückt den Feuerknopf am Joystick, worauf die weiße Umrandung um das Schreibmaschinensymbol verschwindet. Nun lenkt man den Handzeiger etwas nach rechts, so daß er auf die kleine Abbildung eines Druckers zeigt. Nun nochmals den Feuerknopf betätigen und schon bekommt das Druckersymbol einen weißen Rahmen und der Drucker rattert los und druckt den — symbolisch gesprochen — in die Schreibmaschine eingespannten Bogen aus.

Eine Kopie gefällig? Einmal Feuerknopf drücken — Drucker aus, wieder Feuerknopf drücken — die Textseite wird nochmals ausgedruckt. Das läßt sich beliebig fortsetzen, denn die Textseite in der Schreibmaschine bleibt solange erhalten, bis man sie entweder in einer Kartei ablegt oder sie in den Papierkorb wirft.

Letzteres ist genauso einfach wie das Ausdrucken. Mit dem Joystick den Handzeiger auf das Papierkorbsymbol setzen, Feuerknopf betätigen und die Textseite erscheint verkleinert über dem Papierkorb. Falls man es sich anders überlegt hat, kann man jetzt einfach ein anderes Objektsymbol ansteuern, die Textseite landet dann wieder in der Schreibmaschine. Drückt man allerdings ein zweites Mal den Feuerknopf, dann wandert die Textseite — mit einer an »Space Invader« erinnernden Geräuschuntermalung — in den Papierkorb und ist damit endgültig verloren.

Die Floppy ersetzt den Karteikasten

Alle mit der Schreibmaschine erzeugten Textblätter können jedoch auch in einer Kartei archiviert werden. Die Kartei wird vom Hauptmenü — also von der Bildschirmdarstellung des gesamten Schreibtisches her — angewählt. Zur Verfügung stehen drei Karteikästen. Jeder dieser Karteikästen enthält zehn gelbe Indexkarten, deren oberer Rand beliebig beschriftet werden kann und die mit dem Joystick in natürlicher Weise durchgeblättert werden können (Bild 3). Zu jeder gelben Indexkarte wiederum gehören zehn weiße Textkarten, die ebenso wie die gelben Indexkarten beschriftet und durchgeblättert werden.

Das Ablegen einer mit der Schreibmaschine erstellten Textseite geschieht nun sehr einfach und genauso wie in einer realen (Papier-) Kartei. Zunächst wird einer der drei Karteikästen angewählt. Auf dem Bildschirm erscheinen nun die zehn gelben Indexkarten, die man bis zur gewünschten Karte durchblättert und eventuell mit einem Stichwort beschriftet. Mit dem Joystick werden die zur Indexkarte gehörigen weißen Textkarten auf den Bildschirm gebracht und bis zur gewünschten Stelle durchgeblättert. Nun den Feuerknopf drücken und die kleine Abbildung einer Diskette anwählen — und schon wird der Text aus der Schreibmaschine an der ausgewählten Stelle in der Kartei abgelegt.

Um sich einen Textbogen aus der Kartei anzusehen, geht man wie zuvor beschrieben zur gewünschten Stelle in der Kartei, wählt aber dann statt des Diskettensymbols einen kleinen weißen Zettel an. Das entsprechende Karteiblatt wird dadurch in großformatiger Darstellung auf den Schirm gebracht und kann eingesehen oder auch in die Schreibmaschine eingespannt werden.

Da sowohl die gelben Indexkarten als auch die weißen Karteikarten beliebig beschriftet werden können, bietet sich das System für Adressenlisten, Inhaltsverzeichnisse, Terminplanungen oder ähnliche Anwendungen geradezu an.

Eine Diskette kann den Inhalt dreier Karteikästen speichern; das sind immerhin 300 Textbögen, die allerdings nicht alle vollständig beschrieben sein dürfen. Durch Verwendung mehrerer Disketten kann das Speichervermögen der Kartei praktisch beliebig gesteigert werden.

Bei der praktischen Arbeit mit Magic Desk machten sich die doch recht langen Disketten-Zugriffszeiten unangenehm bemerkbar:

Zwischen fünf und zehn Sekunden zum Anwählen eines Karteikastens (nur der reine Diskettenzugriff!) ist noch akzeptabel, aber eine runde Minute zum Abspeichern eines Fünf-Zeilen-Textes zerrt mitunter schon sehr an den Nerven des Benutzers. Allerdings sind diese langen Zugriffszeiten in erster Linie wohl auf den seriellen Datentransport zwischen dem C 64 und dem verwendetem 1541-Laufwerk zurückzuführen.

Magic Desk gibt Hilfestellung

Ein Glanzpunkt des Magic Desk wurde bisher noch nicht erwähnt: Wenn man sich über die Funktion eines Gegenstandes im unklaren ist oder sich sonstwie nicht zu helfen weiß, genügt ein Druck auf die »Commodore«-Taste des C 64 um ein Hilfsmenü aufzurufen. Je nach angewählter Funktion erscheint dabei ein anderes Hilfsmenü, das eine Anzahl von Gegenständen auf den Schirm zeichnet, mit denen es der Benutzer gerade zu tun haben könnte (Bild 4).

In der schon bekannten Art und Weise kann man nun eine dieser Abbildungen mit dem Joystick ansteuern und erhält dann nähere Informationen über den Umgang mit dem entsprechenden Gegenstand. Durch Ansteuerung eines Schildes mit der Aufschrift »EXIT« gelangt man wieder zur gerade aktuellen Funktion zurück.

Insgesamt betrachtet ist das Arbeiten mit dem Magic Desk recht erfreulich, alle Funktionen lassen sich per Joystick leicht und sicher ausführen. Die Einarbeitungszeit ist wesentlich kürzer als bei entsprechenden konventionellen Programmen.

Wer von Magic Desk allerdings einen ausgefeilten Texteditor oder ein komfortables Datenbanksystem erwartet, der wird einigermaßen enttäuscht sein. Magic Desk ist weder das eine noch das andere und soll es auch gar nicht sein. Es ist halt nur ein Schreibtisch — allderdings einer mit recht magischen Zügen. (ev)

Basic Bär (The last one) – Ein Programmgenerator

Sie haben die Idee für ein Programm, Ihnen fehlt aber die Zeit oder die Geduld, sie zu verwirklichen. Hier hilft Ihnen Basic Bär.

Angenommen Sie möchten mit Ihrem C 64 ein komplettes Datenverwaltungssystem einrichten. Wie gehen Sie vor? Programmieren Sie selbst, vielleicht in Basic? Oder kaufen Sie sich ein fertiges Programm? Beide Möglichkeiten haben ihre Vor- und Nachteile. Es gibt aber noch eine dritte Möglichkeit. Sie benutzen einen Programmgenerator.

Ein Programmgenerator ist ein Programm, das nach Ihren Angaben ein Programm erstellt. Dabei müssen Sie selbst kaum programmieren können. Sie geben nur ein sehr allgemeines Ablaufschema des zu erstellenden Programmes ein. Das von Ihnen eingegebene Schema wird nun Schritt für Schritt verfeinert. Nach Ablauf der Generierung steht das erzeugte Programm unabhängig von dem Programmgenerator zu Ihrer Verfügung.

Das gut lesbare Handbuch hat das Format eines Taschenbuches. Die Einarbeitungszeit ist recht kurz, und für einfache Problemlösungen benötigen Sie nicht einmal Basic-Kenntnisse.

Dieser Programmgenerator ist menüorientiert, was bedeutet, daß mit baumartig organisierten Bildschirmmasken alle Möglichkeiten zur Auswahl angeboten werden und Sie Ihre Auswahl nur in Form einer Zahl eingeben brauchen. Diese Dialogform ist recht fehlersicher und hat den Vorteil, daß Sie nach wenigen Stunden kaum noch in das Handbuch sehen müssen.

Die Menütechnik wird überall voll unterstützt, so daß auch das zu erstellende Programm auf diese Weise mit fehlersicheren Eingaberoutinen ausgestattet werden kann.

Die Generierung eines Programmes erfolgt in zwei Schritten. Zuerst geben Sie einen Ablaufplan ein. In einer Frage-Antwortsitzung müssen Sie dann alle Detailfragen, die der Programmgenerator noch benötigt, beantworten.

Der Ablaufplan ist eines der wenigen Dinge, die Sie vorab selbst planen müssen. Ein Ablaufplan im Sinne vom Basic Bär ist eine logische Aufeinanderfolge von Arbeitsschritten, die Sie benötigen, um Ihr geplantes Programm zu verwirklichen. Der Programmgenerator nutzt die Tatsache, daß die einzelnen Verarbeitungsschritte in den meisten Programmen ähnlich sind. Die Ausgabe einer Bildschirmmaske und das anschließende Einlesen von Daten ist ein solcher Arbeitsschritt. Soll zum Beispiel in eine Datei mit dem Namen »Telefonliste« eingelesen werden, so wird dies mit folgendem Befehl umschrieben:

KEYBORD INPUT USING TELEFONLISTE FIELDS

Wenn Sie im Programm eine Abfrage einbauen, und abhängig davon an eine bestimmte Stelle im Ablaufplan verzweigen wollen, müssen Sie etwa folgendes schreiben:

ASK USER »FERTIG ?«: BRANCH IF YES TO LINE X

Wie die Bildschirmmaske und die Datei überhaupt aufgebaut sein soll, und welche Datenfelder denn eingelesen werden sollen, wird wie die Sprungadresse erst in der Codegenerierung abgefragt.

Das generierte Basic-Programm

Wenn der Ablaufplan vollständig eingegeben worden ist, wird Zeile um Zeile der Basic-Code generiert. Das Floppy-Laufwerk kommt dabei kaum zur Ruhe. In dieser Phase beantworten Sie die Fragen nach dem Aufbau von Bildschirmmasken, nach dem Druckbild von Listen, nach Sortierkriterien oder nach den Sprungzielen bei Verzweigungen. Den Aufbau von Bildschirmmasken, Listen und den Ablaufplan können Sie abspeichern, um später auf diese wieder zurückgreifen zu können.

Anschließend werden Ihre gesamten Angaben in Basic übersetzt, ähnlich wie ein Makroassembler Makroanweisungen in Maschinensprache übersetzt.

Ein so erzeugtes Programm ist nicht sehr übersichtlich. Will man später Änderungen vornehmen, muß man sehr viel Zeit aufwenden, um sich in dem Programm zurecht zu finden. Wie zu erwarten war, ist ein mit Basic Bär erstelltes Programm erheblich länger als ein selbst erstelltes. Der Unterschied wird aber erheblich geringer, je komplexer die Anforderungen an das Programm sind. Das Zeitverhalten ist für ein Basic-Programm recht gut. Jedem ernsthaften Anwender sei allerdings ein Basic-Compiler empfohlen, um so das generierte Programm zu beschleunigen.

Dateibearbeitung und andere Funktionen

Alle Routinen zur Dateibearbeitung sind in Basic Bär fest eingebaut. Sie arbeiten allerdings nur mit relativen Dateien. Wer also eine andere Dateiform wünscht, muß die Routinen wie Dateiöffnen, Satzlesen, Positionieren auf einen bestimmten Satz oder Satzschreiben selbst programmieren. Dies ist jedoch ohne weiteres möglich, da innerhalb des Programmgenerators auch sämtliche Basic-Befehle zugänglich sind. Sehr positiv ist zu erwähnen, daß mit einem SORT-Kommando gearbeitet werden kann, mit dem nach bis zu drei Schlüsseln sortiert wird.

Durch eine automatische Ausrichtung und Überprüfung von numerischen Feldern in den Bildschirmmasken und einer automatischen Dokumentation der gesamten Eingaben bei der Generierung werden die von Basic Bär erstellten Programme sehr fehlersicher und wartungsfreundlich.

Basic Bär gehört in die Klasse der guten und durchdachten Dienstprogramme. Der Preis von 425 Mark wird den Anwenderkreis sicherlich einschränken. Aber trotzdem ist dieser Programmgenerator für jene interessant, die sowohl mit dem aufwendigen Selbstprogrammieren als auch mit den starren Programmen aus der Dose unzufrieden sind. (rg)

HES 64 FORTH

Die Programmiersprache Forth ist für immer mehr Mikrocomputer erhältlich, und das nicht ohne Grund. Forth ist eine sehr mächtige Sprache, die in einem gewissen Sinne die positiven Seiten von Basic, Pascal und Assembler in sich vereint. Für den Commodore 64 ist jetzt eine der leistungsfähigsten Forth-Versionen als Steckmodul erhältlich: Das 64 Forth von Human Engineered Software.

```
$$$$$$$$ 64FORTH 1.2 $$$$$$$$
        COPYRIGHT 1983
HUMAN ENGINEERED SOFTWARE  BY T.J.ZIMMER

DECIMAL

OK
SOURCE VLIST
 : VLIST      CONTEXT  @   @   CR   DUP
  ID.   TAB   PFA   LFA   ?TERMINAL
  0BRANCH 13   KEY   CLIT 3   -   0BRA
NCH 12   @   DUP   0=   0BRANCH -35
  DROP   ;S

OK
```

Bild 1. Das 64 Forth enthält einen leistungsfähigen Decompiler

Bei dieser Forth-Version für den C 64 handelt es sich um ein im Sprachumfang stark erweitertes FIG-Forth. Die Abkürzung FIG steht dabei für Forth Interest Group und bezeichnet eine unabhängige Gruppe von Forth-Enthusiasten, die sich die Standardisierung und Verbreitung von Forth zum Ziel gesetzt hat. Bei einer Sprache wie Forth, die vom Anwender praktisch beliebig erweiterbar und veränderbar ist, erscheint ein solcher Standard auch dringend notwendig, weil sonst der Softwareaustausch zwischen den Anwendern fast unmöglich würde. Das 64 Forth hält sich sehr eng an den FIG-Standard, was dem Benutzer unmittelbar zugute kommt: Die meisten für andere Computer entwickelten Forth-Programme sind, solange nicht spezielle Hardware-Eigenschaften ausgenutzt werden, mit minimalen Änderungen sofort auf dem C 64 lauffähig.

Forth zeichnet sich ja generell schon durch einen großen Befehlsvorrat aus. Im 64 Forth wurde jedoch der Grundwortschatz des FIG-Standards nochmals stark erweitert. Über 500 Befehle stehen zur Verfügung. Damit man hier noch den Überblick behält, sind diese Befehle auf vier Vokabulare aufgeteilt: FORTH, EDITOR, SYSTEM und ASSEMBLER. Ein Vokabular wird einfach durch Angabe seines Namens aufgerufen oder aktiviert.

Zu Anfang ist nur das normale Forth-Vokabular zugänglich. Will man längere Forth-Programme oder Texte eingeben, dann ruft man einfach das Editor-Vokabular auf und kann nun alle vorhandenen Befehle zum Editieren von Texten nutzen. Das System-Vokabular enthält Befehle zum Zugriff auf die Betriebssystemebene.

Obwohl Forth-Programme in der Regel um Größenordnungen schneller als Basic-Programme laufen, können doch ab und zu zeitkritische Situationen eintreten. In diesem Fall ruft man das Assembler-Vokabular auf und hat sofort einen 6502-Macro-Assembler zur Verfügung.

Schon bei einem ersten Blick ins Inhaltsverzeichnis des (leider nur englischen) Handbuchs fällt es auf: Das 64 Forth verfügt in Addition zum FIG-Forth-Standard über eine ganze Reihe von Befehlen, um spezielle C 64 — Eigenschaften zu unterstützen.

Mit BGROUND und BORDER werden zum Beispiel die Hintergrund- und die Rahmenfarbe gewählt. Mit NEWSPRITE kann man innerhalb einer Bildschirmmaske sehr komfortabel neue Sprites entwerfen. Spritedaten können in spezielle Sprite-Dateien geschrieben oder daraus gelesen werden. Mit dem Befehl SHOW werden Sprites sichtbar gemacht, mit HIDE verschwinden sie wieder vom Bildschirm. Mit weiteren Befehlen kann man Sprites über den Bildschirm bewegen, einfärben und vergrößern. Natürlich ist auch der Mehrfarbenmodus möglich.

Wer sich schon einmal mit den diversen POKE-Befehlen in Basic abgemüht hat, um ein paar Sprites über den Bildschirm zu bewegen, der wird diese Möglichkeiten des 64 Forth sehr zu schätzen wissen.

Ganz ähnlich verhält es sich auch

Bild 2. Das 64 Forth ist als Steckmodul für den C 64 und den VC 20 erhältlich ▼



KOMFORTABLER ALS BASIC

bei den Toneffekten: Wo man in Basic viele Zeilen mit POKE-Befehlen braucht, um ein paar Noten zu spielen, geht das mit 64 Forth im Klartext und wesentlich übersichtlicher. Der Befehl VOICE1 wählt zum Beispiel den Tongenerator 1 an, mit TRIANGLE, SAWTOOTH, SQUARE oder NOISE wird die Wellenform ausgewählt. Weitere Befehle steuern Frequenz, Hüllkurve, Lautstärke und andere Faktoren. Insgesamt gibt es 40 (!) Befehle zur Steuerung des Synthezisers.

Auch die Ansteuerung externer Geräte macht mit 64 Forth keine Schwierigkeiten. Im SYSTEM-Vokabular gibt es diverse Befehle zur Datenkommunikation mit Drucker, Floppy oder Kassette. Der Befehl PRON ersetzt beispielsweise die Basic-Befehlsfolge »OPEN 1,4:CMD 1«. Um Befehle an die Floppy zu schicken, braucht nicht umständlich ein Kommandokanal eröffnet werden, sondern es reicht der CMD-Befehl. Um zum Beispiel eine Diskette zu formatieren, schreibt man in 64 Forth: CMD N:Name,ID.

Außerdem können die meisten Kernal-Routinen des Betriebssystems einfach mit ihrem Namen aufgerufen werden, wodurch sich eine hohe Flexibilität beim Datenaustausch über den seriellen Bus ergibt.

Bemerkenswert sind die recht komfortablen Testhilfen, die 64 Forth zur Verfügung stellt. Mit den Befehlen TRACE, STEP, EMULATE und CONT kann die Ausführung von Forth-Befehlen überwacht werden. Zusätzlich ist ein Decompiler vorhanden, der mit SOURCE aufgerufen wird und compilierte Forth-Worte wieder zurückübersetzen kann (Bild 1). Derartig umfangreiche Debug-Funktionen sucht man bei anderen Forth-Compilern in der Regel vergeblich.

Im Gegensatz zu der recht einfachen Speicherverwaltung von Basic benutzt Forth das Konzept des virtuellen Speichers. Vereinfacht gesagt bedeutet das die Aufteilung des verfügbaren Speichers in kleine Abschnitte, sogenannte Screens oder Textfelder, auf die über eine Nummer zugegriffen wird. Dabei ist es nicht nötig, daß sich alle Screens zur gleichen Zeit im Hauptspeicher befinden. Die meisten Screens sind daher als relative Datei auf einer Diskette angelegt und werden nur bei Bedarf geladen.

64 Forth bietet dabei noch die Besonderheit, daß eine solche virtuelle Speicherverwaltung auch ohne Floppy ermöglicht wird. Die einzelnen Textfelder werden dabei in der Reihenfolge ihrer Numerierung mit der Datasette wie normale Programme aufgezeichnet. Ein Pufferspeicher von 16 Textfeldern im RAM, begrenzt die Anzahl der nötigen Kassettenoperationen.

Jedes Textfeld kann nun für sich mit dem Texteditor bearbeitet werden. Bei fast allen Forth-Versionen ist zu diesem Zweck nur ein einfacher zeilenorientierter Editor vorhanden. 64 Forth stellt seine Kompatibilität dadurch unter Beweis, daß es ebenfalls über einen solchen umständlich zu bedienenden Zeileneditor verfügt, der ganz normal mit n EDIT aufgerufen wird, wobei n die Nummer des zu editierenden Textfeldes ist.

Der Full-Screen-Editor

Daneben aber gibt es in 64 Forth auch einen »Full-Screen-Editor«, wie er vom Basic her bekannt ist. Man drückt im Edit-Modus die Tastenkombination »Shift« und »INST/DEL«, und es erscheint augenblicklich das gesamte zu editierende Textfeld. Wie von Basic her gewohnt, kann man nun mit Hilfe der Cursourtasten über den gesamten Bildschirm fahren und fehlerhafte Textstellen löschen oder einfach überschreiben.

Die Funktionstasten sind dabei mit diversen hilfreichen Funktionen belegt, zum Beispiel Vorwärts- und Rückwärtstabulator und Suchfunktionen. Auf Tastendruck kann man in das vorhergehende oder nachfolgende Textfeld wechseln.

Als weitere Besonderheit versteht der Editor eine Reihe von »Wordstar«-kompatiblen Control-Funktionen. Statt mit der »CTRL«-Taste werden diese Funktionen jedoch durch gleichzeitiges Drücken der »Commodore«-Taste und eines Buchstabens ausgelöst. Wer also an das Arbeiten mit »Wordstar« gewöhnt ist, wird sich mit diesem Editor ebenfalls gut zurechtfinden.

Ganz hartgesottene Forth-Programmierer, denen soviel Bedienungskomfort schon dekadent erscheint, haben ja noch die Möglichkeit, stattdessen mit dem zeilenorientierten Editor zu arbeiten.

Das zu magere Handbuch ist in Englisch geschrieben, eine deutsche Übersetzung ist zur Zeit nicht geplant.

Auf 155 kleinformatigen Seiten wird eine kurze Einführung in Forth gegeben, die aber dem Anfänger vermutlich nicht allzuviel sagen wird. In kurzen Kapiteln werden dann der Texteditor und die wichtigsten Forth-Befehle erläutert. Es folgen Abschnitte über die Speicherbelegung von Forth, über die I/O-Organisation und über den integrierten 6502-Macro-Assembler. Listings einer Anzahl von Forth-Utilitys und eine Kurzbeschreibung aller Befehle runden das Handbuch ab.

Durchweg alle Kapitel sind aber recht kurz und knapp ausgefallen, so daß man häufig gezwungen ist, entweder in anderen Büchern nachzuschlagen oder einfach herumzuexperimentieren. Immerhin, man findet die wichtigsten Informationen. Dennoch wäre ein deutsches, etwas ausführlicheres Handbuch sicherlich wünschenswert.

Insgesamt gesehen ist das 64 Forth sicherlich eine der besten Forth-Versionen, die derzeit überhaupt für Mikrocomputer erhältlich sind. Deutliche Pluspunkte sind der hervorragende Editor und der umfangreiche Befehlssatz, insbesondere die vielen Befehle für Spritegrafik und Sounderzeugung, die man im Commodore-Basic manchmal schmerzlich vermißt. Der integrierte Assembler ist eine wertvolle Hilfe bei der Lösung zeitkritischer Probleme.

Das 64 Forth kann in Deutschland als Steckmodul (Bild 2) für den C 64 über »Die Forth-Quelle« in 7820 Titisee-Neustadt zum Preis von 198 Mark bezogen werden. Eine Version für den VC 20 ist ebenfalls erhältlich. Leider ist das Modul nach Auskunft der Anbieter auf dem tragbaren SX 64 aufgrund doch vorhandener geringer Hardwareunterschiede nicht lauffähig. (ev)

LODE RUNNER

Die erste Schwierigkeitsstufe von »Lode Runner«



Wer auf seinem Heimcomputer sein Reaktionsvermögen testen will, dem sei Lode Runner empfohlen, ein Spiel mit 150 (!) verschiedenen Spielfeldern.

Das Spiel Lode Runner ist ein Actionspiel aber dennoch kein Schießspiel und wird für Apple II, II+ und IIe, Atari 400/800/XL, VC 20, Commodore 64 sowie IBM PC angeboten.

»Du bist ein gut trainierter Commander im unendlichen Weltraum...« — so verheißungsvoll beginnt die Spielanleitung (natürlich in Englisch abgefaßt). Man muß auch wirklich gut trainiert sein, um alle 150 Spielfelder zu durchlaufen. Ich selbst habe allein sechs Stunden benötigt, um bis zum siebten Spielfeld vorzudringen. Man hat die Möglichkeit, sich ganze 255 eigene Spielfelder zu erstellen und diese auf Kassette oder Diskette abzuspeichern.

Dies ist mit dem im Spiel integrierten Game Generator möglich. Der Generator ist sehr einfach zu bedienen. Obwohl die Bedienung sehr einfach ist, wäre es angebracht, in der Spielanleitung zu erwähnen wie man den Game Generator startet. Ich wollte die Suche nach dem Generator schon aufgeben, als ich zufällig nach dem Start des Spieles die »E«-Taste drückte und sogleich mein eigenes Spielfeld entwerfen konnte.

Doch nun noch einmal zum Spiel selbst.

Der Spieler muß in jedem Feld alle Goldklumpen sammeln. Doch das Einsammeln der Punkte in dem Labyrinth aus Mauern, Leitern und Stangen ist nicht so einfach, da man hierbei von einigen Feinden verfolgt wird. So braucht man schon einiges Geschick und Reaktionsvermögen, um alle Goldklumpen zu sammeln und nicht gefangen zu werden.

Die grafischen und musikalischen Fähigkeiten des VC 20 werden nicht gerade voll ausgeschöpft. Der Spieler bewegt sich zwar fließend aber die Verfolger »hüpfen« nur ruckartig von Kästchen zu Kästchen.

Fazit:

Wer Abwechslung und Spannung liebt, dem sei dieses Spiel empfohlen, da wegen der großen Variationsberichte und der Möglichkeit, selbst kreativ zu werden, die Spielmotivation bestimmt nicht nachläßt.

(Christian Spitzner)



Allen Arcade-Spiel-Freunden ist er sicherlich bekannt...Zaxxon. Der gefährliche Kampfroboter treibt nun schon seit Anfang '84 sein Unwesen auf dem Home-Computer-Markt. Allerdings hört man schon einige Seufzer...ist der Riesenhit doch kleiner als man erwartet hatte? Nun, zur Handlung: Man fliegt mit einem Raumkreuzer über einen Asteroidengürtel, versucht möglichst viel »Feindliches« abzuschießen und steht dann nach einigen Prüfungen vor »Zaxxon«, einem gefährlichen Kampfroboter. Dieser erscheint zwar nur kurz, trotzdem muß man sich mit ihm ein heißes Gefecht liefern. Nach dieser Konfrontation beginnt alles wieder von vorne. — Nur etwas schwerer. Die starke pseudo-3D-Grafik wertet das Spiel zwar auf, jedoch hilft selbst die beste Grafik nicht, über Schwächen hinwegzutäuschen, die nach einigen Spielen deutlich werden: So wird es dem Spieler leicht langweilig, wenn er 5 bis 6 mal immer wieder dieselbe Szene überfliegt und sich nichts ändert, außer dem Schwierigkeitsgrad. Generell ist zu sagen, daß a) die Erwartungen der Käufer zu hoch gesteckt waren und b) daß Zaxxon diese hohen Ansprüche nicht erfüllen kann — und das schon gar nicht auf Dauer! Alles in allem, Zaxxon ist nicht das, was man so hört, wird aber sicher bei den Weltraumabenteuern einen gehobenen Rang einnehmen, den er sich trotz alledem verdient hat.

(Oliver v. Quadt)

Zaxxon: Eine typische Spielszene



Auch der Blick von oben ist möglich

FLIGHT II

Fast wie richtiges Fliegen

Wenn Sie auch zu den Zeitgenossen gehören, denen das Fliegen nur auf dem Passagiersitz eines Linienflugzeugs vergönnt ist, dann gibt es etwas Neues für Sie und Ihren Commodore 64.

Der Flight-Simulator II von Sublogic im Vertrieb der Lucius Computer-Programme (190 Mark) stellt ohne Übertreibung alle bekannten Flug-Programme in den Schatten.

Das Fluggerät ist eine Piper Pa 28-181 Archer II. Ihre Instrumente und Flugeigenschaften werden täuschend echt simuliert. Über 80 Flughäfen kann der Computer-Pilot in den USA ansteuern. Ein riesiges Terrain, das nur die wenigsten überfliegen werden. Die relativ realistische Abbildung des Geländes ist allein schon ein Genuß für das Auge.

Nun zum Fluggeschehen. Je nach Können und vor allem Mut ist es möglich, zwischen drei verschiedenen Schwierigkeitsgraden zu wählen: Dem »realistic«, dem »easy« und dem »slew« Modus. Die höchsten Ansprüche an die Fähigkeiten des Piloten stellt der Realflug. Hier ist vom Starten des Motors über den Kampf mit den verschiedensten Handicaps und Defekten bis zum sensiblen Flugverhalten alles vorhanden. In den vollen Genuß der Landschaft und der Flugumgebung kommt man am besten durch Anwählen des »slew« Modus. Das Flugzeug fliegt dann wie ein Hubschrauber.

Zu Beginn der fliegerischen Laufbahn ist es sinnvoll, der ausgezeichneten (noch englischen) Anleitung zu folgen. Dadurch erspart man sich so machen unfreiwilligen »Crash«.

Der Horizont und die Landschaftsmerkmale bewegen sich beim Fliegen naturgetreu, entsprechend der jeweiligen Flugposition über den Bildschirm. Aber Vorsicht; nicht die Instrumente vergessen! Schnell kann es geschehen, daß man übersteuert oder zu niedrig fliegt.

Starten möchte ich in New York. Dazu sind die Koordinaten des Abflugortes im Editmodus einzugeben; in diesem Fall ist das 17070/20990. Nach kurzen Diskettenmanövern, die leider im ganzen Spiel immer wieder notwendig werden, erscheint New York auf dem Bildschirm. Vorsichtig die Geschwindigkeit erhöhen, leicht nach oben ziehen, sanft eine Rechtskurve einleiten, nur nicht übersteuern und immer den Höhenmesser im Auge behalten — ein Kommando folgt auf das nächste. Vor mir erscheint das World Trade Center mit seinen beiden Türmen. Der Abenteurer in mir wird wach. Etwas tiefer noch, und da sind sie auch schon, immer größer und bedrohlicher werdend. Als ich zwischen den Türmen durchfliege, wage ich einen Blick nach unten und was ich sehe läßt meinen Atem stocken: Ich bin zu tief! Dies waren meine letzten Gedanken. Als ich »aufwachte«, stand nur noch »crash« auf dem Bildschirm.

Das Edit-Modus

Nicht unerwähnt soll auch das auswählbare Actionspiel im Flight II bleiben. Nicht etwa, daß Flight II ohne das Luftkampfspiel »World War I Ace« kein vollständiges Programm wäre. Im Gegenteil, beide Programme könnten auch separat verkauft werden. Die Aufgabe des Piloten besteht beim »World War I Ace« darin, feindliche Stellungen, Depots und Fabriken zu beschießen. Das Flugzeug ist dazu mit Maschinenkanonen und Bomben bewaffnet. Damit der Angriff realistischer wird, wehrt sich der Feind mit sechs äußerst aufdringlichen Jagdfliegern. Diese sind nur durch waghalsige Flugmanöver und intensiven Einsatz der Bordkanone abzuschütteln.

Es ist unbestreitbar, auch dieses Spiel begeistert, mich persönlich mehr durch die gelungene Grafik und Animation, als durch die Spielidee. Wer aber gerne als »Richthofen« durch die Lüfte jagen möchte, wird sicher einige seiner bisherigen »Actionspiele« zumindest vorläufig beiseitelegen.

Insgesamt, wobei der Flugsimulator der wichtigere Teil ist, kenne ich kein Programm, das den Flight II in bezug auf Spielmotivation, Grafik, Beeinflußbarkeit der Handlung und Abwechslungsreichtum erreichen könnte. Am besten gefällt mir aber, daß mit Flight II ein Unterhaltungsprogramm geschaffen wurde, das sicher nicht nur die jüngere Generation begeistert. (Arnd Wängler)

Der Softwarekatalog für Ihre Programme

Wie finden Sie Ihre auf Disketten gespeicherten Programme ohne langes Suchen? Abhilfe in diesem Punkt schafft dieses Programm, mit dem Sie Ihren eigenen Softwarekatalog erstellen können.

Das Programm für den Commodore 64 mit Diskettenlaufwerk und Epson-Drucker RX-80 mit Data-Becker Interface (simuliert einen Commodore-Drucker auf dem RX-80) erlaubt die Verwaltung der auf Diskette gespeicherten Programme. Man legt nur eine Diskette ein, dann lädt der Computer das Inhaltsverzeichnis der Diskette und druckt es auf dem Bildschirm aus. Nun tippt man nur noch »J« für Ja oder »N« für Nein ein, ob der Computer den Programmnamen in seine Liste aufnehmen soll oder nicht. Danach gibt man ein, unter welcher Kategorie das Programm eingeordnet werden soll. Zur Auswahl stehen allgemeine, Spiel-, Hilfs- und Sprachprogramme.

Im Menü kann man zwischen dem alphabetischen Sortieren der Liste, Speichern der Liste und dem Auflisten und Löschen von Programmen wählen. Beim alphabetischen Sortieren werden zunächst alle Namen, die zweimal gespeichert sind, gelöscht. Dann wird die Liste nach ASCII-Wertigkeit geordnet. Das Abspeichern der Liste erfolgt in eine sequentielle Datei. Auf ein »Random-File« wurde verzichtet, da sonst das Programm um einiges länger und langsamer geworden wäre. Bei »Programme löschen« kann man Programmfiles löschen und umbenennen. Will man umbenennen, so gibt man zuerst den alten, noch gespeicherten Namen, dann den neuen Namen des Programmes ein.

Beim Auflisten von Files kann man die Programme entweder auf dem Bildschirm oder dem Drucker ausgeben. Dabei kann man alle oder nur einzelne Kriterien auflisten.

Zur besseren Übersicht und zum schöneren Aussehen werden die Farben Rot, Gelb, Blau, Schwarz und Weiß benutzt. Diese werden in Control-Strings abgespeichert, da sonst das Programmlisting sehr unübersichtlich und beim Eintippen schwierig zu lesen wäre.

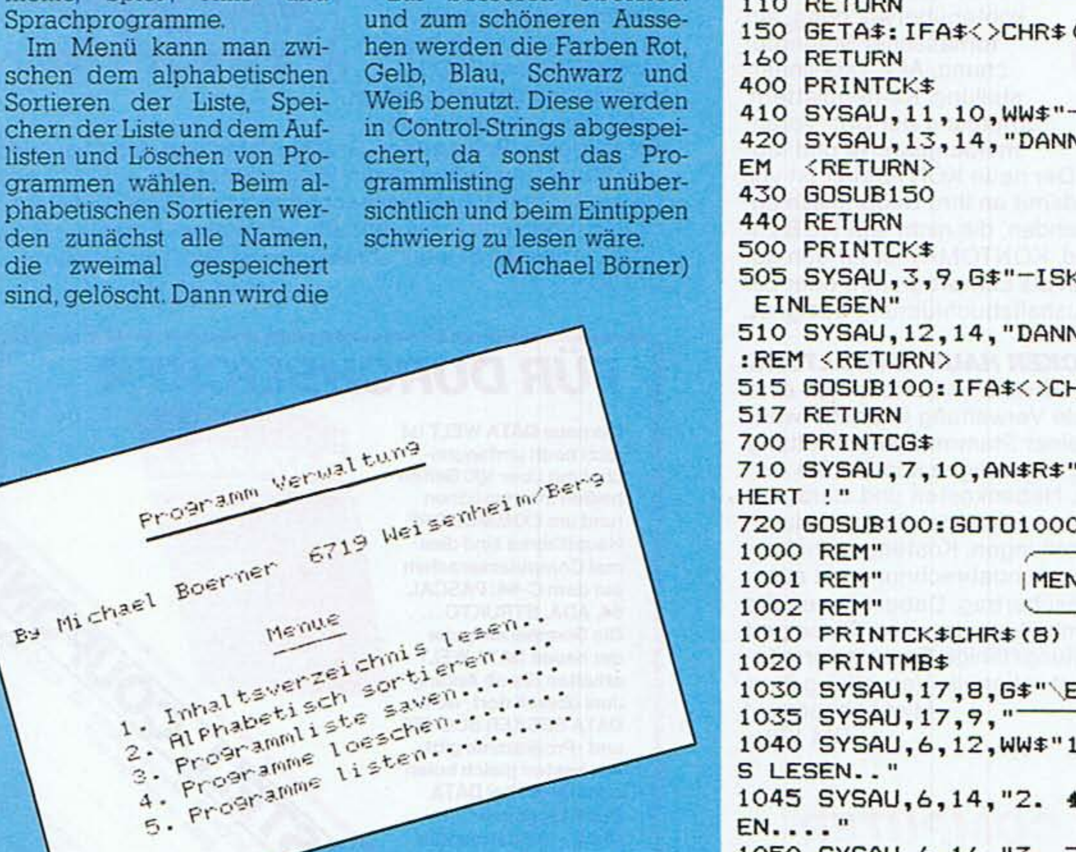
(Michael Börner)

```
EIN AUSDRUCK VON PROGRAMMFILES SIEHT FOLGENDERMASSEN AUS

          PROGRAMMLISTE COMMODORE 64

 1)  ADRESSVERWALTUNG   ALLG.-PRG.     12)  80 ZEICHEN KARTE   HILFS-PRG.
 2)  DEMO 2             ALLG.-PRG.     13)  C000 MONITOR       HILFS-PRG.
 3)  KARTEIVERWALTUNG   ALLG.-PRG.     14)  FUNKTIONEN PLOTT   HILFS-PRG.
 4)  MEMORYMAP-64       ALLG.-PRG.     15)  MC-SAVE            HILFS-PRG.
 5)  ATOMKRAFTWERK 64   SPIEL          16)  MEMORY-CHR$-LIST   HILFS-PRG.
 6)  CAVERN OF RICHES   SPIEL          17)  PRO-VERWALTUNG     HILFS-PRG.
 7)  GOLDGRAEBER        SPIEL          18)  RE-NEW             HILFS-PRG.
 8)  MONDLANDUNG        SPIEL          19)  TURBO TAPE         HILFS-PRG.
 9)  MONOPOLY           SPIEL          20)  FORTH              SPRACH-PRG.
10)  SUPERHIRN          SPIEL          21)  LOGO               SPRACH-PRG.
11)  6000 MONITOR       HILFS-PRG.
```

Beispielausdruck



Hauptmenü

```
99 GOTO10000
100 GETA$:IFA$=""THEN100
110 RETURN
150 GETA$:IFA$<>CHR$(13)THEN150
160 RETURN
400 PRINTCK$
410 SYSAU,11,10,WW$"-ISKETTE EINLEGEN"
420 SYSAU,13,14,"DANN "AN$R$"<_¬| ,_/>":R
EM <RETURN>
430 GOSUB150
440 RETURN
500 PRINTCK$
505 SYSAU,3,9,G$"-ISKETTE MIT /AMENLISTE
 EINLEGEN"
510 SYSAU,12,14,"DANN  "AN$BL$"<_¬| ,_/>"
:REM <RETURN>
515 GOSUB100:IFA$<>CHR$(13)THEN515
517 RETURN
700 PRINTCG$
710 SYSAU,7,10,AN$R$"KEINE NAMEN GESPEIC
HERT !"
720 GOSUB100:GOTO1000
1000 REM"
1001 REM"         |MENUE|
1002 REM"
1010 PRINTCK$CHR$(8)
1020 PRINTMB$
1030 SYSAU,17,8,G$"\ENUE"
1035 SYSAU,17,9,"─────"
1040 SYSAU,6,12,WW$"1. ⌐NHALTSVERZEICHNI
S LESEN.."
1045 SYSAU,6,14,"2. ♠LPHABETISCH SORTIER
EN...."
1050 SYSAU,6,16,"3. ¬ROGRAMMLISTE SAVEN.
......"
1060 SYSAU,6,18,"4. ¬ROGRAMME LOESCHEN..
```

Listing »Software-Katalog«

```
......"
1070 SYSAU,6,20,"5. PROGRAMME LISTEN....
......"
1100 GOSUB100
1110 IFA$<>"1"ANDA$<>"2"ANDA$<>"3"ANDA$<
>"4"ANDA$<>"5"THEN1100
1120 A=VAL(A$):ONAGOTO3000,3600,2500,700
0,5000
2000 REM"
2001 REM"          |DATEN AUS DATEI LESEN|
2002 REM"
2010 GOSUB500
2020 CLOSE2:OPEN2,8,2,"NAMENLISTE,S,R"
2030 INPUT#2,PO
2040 IFPO=0THEN2100
2050 FORN=1TOPO-1
2060 INPUT#2,WR$(N)
2070 NEXTN
2080 CLOSE2:GOTO1000
2100 REM"
2101 REM"          |NEUE DATEI ERDEFFNEN|
2102 REM"
2110 SYSAU,9,20,AN$R$"/EUE -ATEI ERDEFFN
EN ?"
2120 GOSUB100:IFA$<>"J"ANDA$<>"N"THEN212
0
2130 IFA$="N"THENGOTO2000
2140 PO=1:CLOSE2
2150 OPEN2,8,2,"§:NAMENLISTE,S,W"
2160 PRINT#2,PO
2170 CLOSE2
2200 GOTO1000
2500 REM"
2501 REM"          |NAMEN ABSAVEN|
2502 REM"
2505 IFPO<2THENGOTO700
2510 GOSUB500
2520 OPEN2,8,2,"§:NAMENLISTE,S,W"
2530 PRINT#2,PO
2540 FORN=1TOPO-1
2550 PRINT#2,WR$(N)
2560 NEXT
2570 CLOSE2:GOTO1000
3000 REM"
3001 REM"          |INHALTSVERZEICHNIS LADEN|
3002 REM"
3005 R=0
3010 GOSUB400
3015 PRINTCG$
3020 SYSAU,3,3,G$"EINLESEN DES INHALTSVE
RZEICHNISSES":PRINT:PR=PO
3025 CLOSE15:OPEN15,8,15,"IO":CLOSE2:OPE
N2,8,0,"$"
3030 FORT=0TO12:GET#2,A$,B$:NEXT
3035 GET#2,A$,A$:IFSTTHENCLOSE2:GOTO3120
3040 GET#2,A$,A$
3050 GET#2,A$:IFA$=""THEN3035
3060 IFA$<>CC$THEN3050
3070 C$="":R=R+1:C$(R)="":P$(R)="":FL=0
3080 FORT=0TO16:GET#2,A$:IFA$<>CC$ANDFL=
0THENC$(R)=C$(R)+A$:NEXT:GOTO3100
3090 FL=1:NEXT
3100 IFINT(R/2)=R/2THENPRINT:PRINTS$C$(R
-1)TAB(20)C$(R);
3105 PRINTTAB(37)P$(R)G$;
3110 GET#2,A$,P$(R):GOTO3050
3111 REM"
3112 REM"          |NAMEN IN DATEI UEBERNEHME
N ?|
3113 REM"
3120 PRINTCG$:PRINTG$"UEBERNEHMEN ?"
3121 IFR=0THEN1000
3125 FORT=1TORSTEP2:PRINT:PRINTC$(T)TAB(
17)P$(T)TAB(20)C$(T+1);
3130 PRINTTAB(37)P$(T+1);:NEXT:POKE198,0
3135 R2=1
3150 FORT=1TOR:P=T*20+55396:FORTT=0TO19
3160 POKEP+TT,0:NEXT
3170 GOSUB100
3180 IFA$<>"N"ANDA$<>"J"THEN3170
3190 IFA$="J"THENWR$(PO)=" "+C$(T):PO=PO
+1:F1=1:GOTO3300
3200 IFA$="N"THENF1=12:GOTO3300
3300 FORTT=0TO18:POKEP+TT,F1:NEXT:NEXT
3400 REM"
3401 REM"          |WELCHE SPIELART ?|
3402 REM"
3403 IFPO=PRTHEN1000
3405 PRINTCG$
3408 SYSAU,1,1,AN$R$"G"AU$G$"AME/"AN$R$"
H"AU$G$"ILFSPRG/":
3410 PRINTAN$R$"S"AU$G$"PRACHPRG/"AN$R$"
A"AU$G$"LLGEMEIN"
3412 PRINT:PRINT
3415 FORN=PRTOPO-1
3417 IFLEN(WR$(N))>19THEN3425
3420 W=LEN(WR$(N)):FORM=WTO18:WR$(N)=WR$
(N)+" ":NEXTM
3425 PRINTRIGHT$(WR$(N),LEN(WR$(N))-1);
3430 GOSUB100:IFA$<>"G"ANDA$<>"A"ANDA$<>
"S"ANDA$<>"H"THEN3430
3440 IFA$="G"THENPRINTTAB(20)"SPIEL
3450 IFA$="A"THENPRINTTAB(20)"ALLGEMEIN
3460 IFA$="S"THENPRINTTAB(20)"SPRACH
3470 IFA$="H"THENPRINTTAB(20)"HILFE
3480 B$=MID$(WR$(N),2,LEN(WR$(N))):A$=A$
+B$:WR$(N)=A$:NEXT
3500 PRINT:PRINT:PRINTAN$R$"OK"AU$" ?"
3510 GOSUB100:IFA$<>"J"ANDA$<>"N"THEN351
0
3520 IFA$="N"THEN3400
3525 GOTO1000
3600 REM"
3601 REM"          |FILES LOESCHEN FALLS DOPP
ELT|
3602 REM"
3603 PRINTCG$
3604 SYSAU,13,11,R$"BITTE WARTEN !"
3605 IFPO=2THEN1000
3610 FORN=1TOPO-1
3620 FORM=1TOPO-1:IFM<>NTHENIFWR$(N)=WR$
(M)THENGOTO3700
3630 NEXTM:NEXTN:GOTO3800
3700 FORQ=MTOPO-1
3710 WR$(Q)=WR$(Q+1)
3720 NEXTQ
3730 PO=PO-1:WR$(PO)="":WR$(PO+1)="":GOT
O3630
3800 REM"
3801 REM"          |ALPHABETISCH SORTIEREN|
3802 REM"
3805 PRINTCG$
3810 SYSAU,13,11,R$"BITTE WARTEN !!"
3820 FORO=1TOPO-2
3830 A$=WR$(O):B$=WR$(O+1)
3840 IFA$<B$THENNEXT:GOTO3900
3850 WR$(O+1)=A$:WR$(O)=B$
3860 NEXTO
3900 FORN=1TOPO-2
3910 IFWR$(N)<WR$(N+1)THENNEXT:GOTO1000
3920 GOTO3820
5000 REM"
5001 REM"          |PROGRAMME LISTEN|
5002 REM"
5005 IFPO<2THENGOSUB700
5006 IFDRTHENCLOSE1
5008 PRINTCK$
5010 SYSAU,12,4,BL$"PROGRAMME LISTEN"
```

```
5015 SYSAU,12,5,"───────────────────"G$
5020 SYSAU,10,7,"1. ALLE TROGRAMME..."
5030 SYSAU,10,9,"2. NUR ♠LL⌐\. TRG..":R
EM       ALLGEM. PRG
5040 SYSAU,10,11,"3. NUR |L_♥-TRG....":
REM       HILFS-PRG
5050 SYSAU,10,13,"4. NUR ♥⌐⌐L-.......":
REM          SPIELE
5060 SYSAU,10,15,"5. NUR ♥⌐L♠-|⌐/.....":
REM       SPRACHEN
5070 SYSAU,11,17,"AUF "AN$S$"-"AU$G$"RUC
KER: ";:REM DRUCKER
5080 IFDRTHENPRINT"_"BL$"JA  "
5085 IFNOTDRTHENPRINT"BL$"NEIN"
5090 SYSAU,10,20,R$"<_⌐| _/>"G$" ZUM "S
$"\ENUE":REM <RETURN> , MENUE
5100 GOSUB100:E$=A$
5110 IFE$="1"ORE$="2"ORE$="3"ORE$="4"ORE
$=CHR$(13)ORE$="D"ORE$="5"THEN5120
5115 GOTO5100
5120 IFE$=CHR$(13)THENDR=0:CLOSE1:GOTO10
00
5130 IFE$="D"THENDR=NOTDR:GOTO5000
5140 IFNOTDRTHEN5200
5150 CLOSE1:OPEN1,4,1:PRINT#1,CHR$(14);
5160 PRINT#1,"       PROGRAMMLISTE COMMO
DORE 64
5170 CLOSE1:OPEN1,4
5180 PRINT#1,"       ────────────────────
5190 CMD1
5200 IFE$="2"THENE$="A"
5205 R=0
5210 IFE$="3"THENE$="H"
5220 IFE$="4"THENE$="G"
5230 IFE$="5"THENE$="S"
5240 IFE$<>"1"THEN5300
5250 IFNOTDRTHEN5300
5260 GOSUB5800
5300 W=1:IFNOTDRTHENPRINTCG$
5310 FORN=1TOPO-1
5330 IFE$="1"THEN5370
5340 IFE$<>LEFT$(WR$(N),1)THENNEXT:GOTO5
450
5370 GOSUB5900
5380 IFINT(W/2)=W/2ANDDRTHENPRINT
5430 W=W+1
5440 NEXT
5450 IFW<>1THENGOSUB100
5470 GOTO5000
5750 RETURN
5800 LO=INT((PO-1)/2):IFINT(LO/2)=LO/2TH
ENLO=LO+1
5805 FORT=1TO(PO-1)/2:N=T:W=N
5810 GOSUB5900
5815 N=N+LO:W=N
5820 GOSUB5900
5825 N=N-LO:W=N
5835 NEXT
5840 IFINT(PO/2)=PO/2THENN=N+1:W=W+1:GOS
UB5900
5850 PRINT#1
5860 DR=0:GOTO5000
5900 Z$=LEFT$(WR$(N),1)
5901 IFW<100THENPRINT" ";
5902 IFW<10THENPRINT" ";
5905 W$=STR$(W):PRINTG$W$")"TAB(3)RIGHT$
(WR$(N),LEN(WR$(N))-1);
5910 IFZ$="A"THENPRINTBL$"ALLG.-PRG.
";
5920 IFZ$="G"THENPRINTWW$"SPIEL
";
5930 IFZ$="H"THENPRINTR$"HILFS-PRG.  "
;
5940 IFZ$="S"THENPRINTS$"SPRACH-PRG.  "
;
5950 IFNOTDRTHENIFW/23=INT(W/23)ANDNOTDR
THENGOSUB100:PRINTCG$;
5965 IFNOTDRTHENPRINT
5970 RETURN
7000 REM"       ┌──────────────────────┐
7001 REM"       |PROGRAMME LOESCHEN|
7002 REM"       └──────────────────────┘
7003 IFPO<2THENGOSUB700
7004 PRINTCK$
7005 SYSAU,11,7,BL$"TROGRAMME LOESCHEN"
7008 SYSAU,11,8,"──────────────────"
7010 SYSAU,9,11,G$"1.  -ILES UMBENENNEN.
.":REM  FILES
7020 SYSAU,9,13,"2.  -ILES LOESCHEN...."
7040 SYSAU,11,16,R$"<_⌐| _/>"G$" ZUM "S$
"\ENUE":REM <RETURN>,MENUE
7050 GOSUB100:IFA$<>"1"ANDA$<>"2"ANDA$<>
CHR$(13)THEN7050
7060 IFA$=CHR$(13)THEN1000
7070 U$="":N$=""
7080 A=VAL(A$):ONAGOTO7100,7500
7100 GOSUB7400
7110 PRINT:U$="":INPUTU$:IFU$=""THEN7000
7120 IFLEN(U$)>16THEN7100
7125 FORN=LEN(U$)TO17:U$=U$+" ":NEXT
7130 GOSUB7400:SYSAU,14,10,S$U$G$
7140 GOSUB7450
7150 N$="":INPUTN$:IFN$=""THEN7000
7160 IFLEN(N$)>16THEN7130
7170 GOSUB7400:SYSAU,14,10,S$U$:GOSUB745
0:SYSAU,14,14,S$N$
7180 SYSAU,1,18,R$" OK ?"
7190 FORN=LEN(N$)TO17:N$=N$+" ":NEXT
7200 :GOSUB100:IFA$<>"J"ANDA$<>"N"THEN72
00
7210 IFA$="N"THEN7100
7220 FORN=1TOPO-1
7230 IFU$=RIGHT$(WR$(N),LEN(WR$(N))-1)TH
ENWR$(N)=LEFT$(WR$(N),1)+N$:GOTO7000
7240 NEXT
7250 PRINT:PRINT:PRINT"  "U$" NICHT GESPE
ICHERT !"
7260 FORN=1TO4000:NEXT
7270 GOTO7000
7400 REM"       ┌──────────────────┐
7401 REM"       |FILES UMBENENNEN|
7402 REM"       └──────────────────┘
7405 PRINTCG$
7410 SYSAU,12,5,BL$"FILES UMBENENNEN:"
7420 SYSAU,8,9,G$"ZU LOESCHENDES PROGRAM
M:"
7430 RETURN
7450 SYSAU,8,13,G$"NEUER NAME DES PROGRA
MMS:"
7460 RETURN
7500 REM"       ┌────────────────┐
7501 REM"       |FILES LOESCHEN|
7502 REM"       └────────────────┘
7510 PRINTCG$
7520 GOSUB7800
7530 N$="":INPUTN$:IFN$=""THEN7000
7540 IFLEN(N$)>16THEN7500
7550 PRINTCG$
7554 GOSUB7800:SYSAU,13,14,S$N$
7556 SYSAU,2,18,R$"OK ?"
7560 GOSUB100:IFA$<>"J"ANDA$<>"N"THEN756
0
7570 IFA$="N"THEN7500
7575 FORN=LEN(N$)TO17:N$=N$+" ":NEXT
7580 FORN=1TOPO-1:IFRIGHT$(WR$(N),LEN(WR
$(N))-1)<>N$THENNEXTN:GOTO7600
7590 FORT=NTOPO-1:WR$(T)=WR$(T+1):NEXT:P
O=PO-1:GOTO7000
```

```
7600 PRINT:PRINT:PRINT" "N$" NICHT GESPE
ICHERT !"
7700 FORN=1TO4000:NEXT:GOTO7000
7750 SYSAU,10,21,R$U$" NICHT GESPEICHERT
 !"
7760 FORN=1TO4000:NEXT:GOTO7000
7800 SYSAU,13,8,BL$"FILES LOESCHEN"
7810 SYSAU,13,9,"──────────────────"
7820 SYSAU,0,12,G$"WELCHES PROGRAMM SOLL
 GELOESCHT WERDEN ?"S$:PRINT
7830 RETURN
10000 REM"          ╭───────────────╮
10001 REM"          |INITIALISIERUNG|
10002 REM"          ╰───────────────╯
10010 MB$="[SHIFT+CLR-HOME][CRSR-DOWN]          [CTRL+7]ROGRAMM XERWAL
TUNG          "
10020 MB$=MB$+"                ──────────
"
10030 MB$=MB$+"[CRSR-DOWN]              [CTRL+3]|Y \ICHAEL |O
ERNER  6719 OEISENHEIM/|ERG        "
10040 REM"[SHIFT+CLR-HOME]"=SHIFT+CLR-HOME; "[CRSR-DOWN]"=CRSR-DO
WN; "[CTRL+7]"=CTRL+7; "■"=CTRL+1; "[CTRL+3]"=CTRL+3
10050 DIMC$(70):DIMP$(70):CC$=CHR$(34):R
EM VARIABLEN FUER INHALTSVERZEICHNIS
10060 DIMWR$(500):REM        PROGRAMM-FIL
E $
```

```
10070 CG$="[SHIFT+CLR-HOME]"+CHR$(142):REM CLEAR-GROSS
10080 CK$="[SHIFT+CLR-HOME]"+CHR$(14):REM  CLEAR-KLEIN
10090 R$=CHR$(28):REM             ROT$
10100 G$=CHR$(158):REM            GELB$
10110 BL$=CHR$(31):REM            BLAU$
10120 S$=CHR$(144):REM            SCHWARZ$
10130 WW$=CHR$(5):REM             WEISS$
10140 AN$=CHR$(18):REM            INVERS AN$
10150 AU$=CHR$(146):REM           INVERS AUS$
10160 POKE53281,12:REM            HINTERGRUNDF
ARBE=GRAU 2
10170 POKE53280,11:REM            RAHMENFARBE
    =GRAU 1
10180 POKE650,128:REM             AUTOREPEAT A
UF ALLEN TASTEN
10190 AU=49152:REM                STARTARESSE
FUER HILFSROUTINE
10200 FORN=AUTOAU+25:REM HILFSROUTINE
10210 READWE:POKEN,WE:NEXT
10220 DATA32,253,174,32,158,183,138
10230 DATA72,32,253,174,32,158,183
10240 DATA104,168,24,32,240,255,32
10250 DATA253,174,76,164,170
11111 GOTO2000
READY.
```

Listing Softwarekatalog (Schluß)

Leserservice:

Die Alternative zum mühsamen Abtippen

Alle Listings, die im 64'er-Magazin veröffentlicht werden, gibt es künftig auch auf Datenträgern. Zusätzlich bieten wir auch Autoren besonders langer Basic- oder Maschinenprogramme eine neue Möglichkeit an, ihre Arbeiten zu veröffentlichen. Solche Programme werden künftig ebenfalls über den Leserservice angeboten.

Viele Leser haben uns gefragt, ob die veröffentlichten Listings auch auf Datenträgern angeboten werden. Mit dem Leserservice kommen wir diesem Wunsch nach. Es werden, nach VC 20 und C 64 getrennt, jeweils alle in einer 64'er-Ausgabe enthaltenen Listings auf Kassette angeboten.

Eine Diskettenversion dieser Programme ist in Vorbereitung und wird in einer der nächsten Ausgaben angeboten. Mit Erscheinen dieses Heftes sind die Kassetten mit den Programmen der Ausgaben 4 und 5 des 64'er-Magazins verfügbar. Diese Kassetten können mit der beigehefteten »Buchladen-Software-Bestellkarte« unter folgenden Nummern bestellt werden.

Ausgabe 4,
C 64-Programme: CB 007
Ausgabe 4,
VC 20-Programme: VC 006
Ausgabe 5,
C 64-Programme: CB 008
Ausgabe 5,
CV 20-Programme: VC 007

Diese Kassetten kosten pro Stück 29,90 Mark. Kassetten mit den Listings der Ausgaben 6 und 7 werden ab Erscheinen von Heft 8 angeboten.

Der Extra-Service

Oft erreichen die Redaktion interessante Programme, die leider zu lang sind, um sie im Heft als Listing zu veröffentlichen. Oder sie sind in Assembler geschrieben. Viele dieser Programme brauchen den Vergleich mit teuren kommerziellen Produkten nicht zu scheuen.

Solche Programme veröffentlichen wir ***zusätzlich*** zu unserem bisherigen Listingangebot in einer neuen Form: Sie werden im Heft vorgestellt. Darin finden Sie die Programmbeschreibung, die Bedienungsanleitung, kurz alles, was Sie für eine komplette Dokumentation benötigen. Das Programm selbst wird auf Kassette angeboten. Auch für diesen Service sind die Diskettenversionen in Vorbereitung. Auf den folgenden Leserservice-Seiten finden Sie die ersten drei Programme: ein Lernprogramm für russische Vokabeln, eine Superversion des Spiels »Space Invaders« und ein Programm, das einen Geldspielautomaten simuliert und über ausgezeichnete Grafik verfügt.

Auch diese Programme sind über die »Buchladen-Software-Bestellkarte« erhältlich. Die Bestellnummer ist jeweils am Ende der Dokumentation aufgeführt. Der Preis für diese Programme beträgt 29,90 Mark.

Die Preise für die Leserservice-Kassetten und -Disketten müssen die Kosten decken — wir sind aber bemüht, sie so niedrig zu halten, daß der Leserservice auch wirklich ein Service ist.

Ihre 64'er-Redaktion

Russische Vokabeln — das Lernprogramm mit den zwei Zeichensätzen

Beim Erlernen einer Fremdsprache ist das Lernen von Vokabeln, Redewendungen und Satzkonstruktionen eine ziemlich unangenehme Aufgabe.

Ein Computer mit einem Vokabellernprogramm kann einem das Lernen selbst nicht abnehmen. Er kann aber ohne große Mühe ein »elektronisches« Vokabelheft führen, verwalten, ausdrucken, zum Abfragen verwenden, das Lernen durch gezieltes Wiederholen noch nicht beherrschter Wortpaarungen verbessern helfen und das Lernen durch den Umgang mit dem Computer interessanter erscheinen lassen.

Viele Vokabellernprogramme wurden schon veröffentlicht. Das im folgenden vorgestellte Programm für den Commodore 64 zum Lernen russischer Vokabeln hat jedoch die folgenden herausstechenden Merkmale:
1. Erweiterung des Zeichensatzes auf Bildschirm und Drucker, um die kyrillischen Buchstaben darstellen zu können
2. ein hoher Bedienungskomfort, denn Auswahl, Eingabe, Abfragen und so weiter erfolgen menü- beziehungsweise maskengesteuert.

Für diejenigen, die an der russischen Sprache nicht interessiert sind, kann das Programm vielleicht als Anregung dienen, denn viele Sprachen haben ja in ihrer Schrift Buchstaben, die nicht im normalen Zeichensatz der Computer enthalten sind (zum Beispiel Griechisch, Französisch).

Das Programm »Russische Vokabeln« kann maximal fünfzig Vokabeln im Speicher des Computers aufnehmen und zum Abfragen, Drucken, Ändern und so weiter verwenden. Eine Vokabel besteht aus fünf Blöcken:

— Russisches Wort, eventuell mit grammatikalischen Erläuterungen,
— deutsche Übersetzung (en),
— Bemerkungen,
— ein oder mehrere russische Satzbeispiele und
— die deutsche Übersetzung der Satzbeispiele.

Jeder der fünf Blöcke kann maximal drei Zeilen mit maximal 39 Zeichen fassen. Die Mindesteingabe ist eine russische Vokabel mit deutscher Übersetzung. Da das Lernen von »nackten« Wortpaaren später oft zu einer fehlerhaften Anwendung des Wortes im Satz führt, ist es zweckmäßig, auch ein kleines Satzbeispiel mit Übersetzung zu lernen. Dies ist mit der in diesem Programm vorhandenen Vokabelstruktur möglich. Der Block »Bemerkungen« kann beliebige Informationen enthalten. Sie werden beim Abfragen der Vokabeln immer angezeigt, ganz egal in welcher Richtung abgefragt wird. Man kann hier zum Beispiel die Wortart eingeben.

Die maximal 50 Vokabeln im Speicher des Computers können auf einer Diskette unter einer Nummer (zwischen 0 und 999) abgespeichert werden. Die Abspeicherung erfolgt als sequentielle Datei mit dem Namen »RUSS.VOK.nnn« mit nnn als Dateinummer am Ende der Eingabe beziehungsweise Änderung von Vokabeln.

Das Programm läuft unter Simons Basic und folgender Hardware-Konfiguration: Commodore-64, Floppy 1541 und Drucker 1525 oder Seikosha GP-100 VC.

Nach dem Laden und Starten von Simons Basic kann das Programm geladen und

Variable	Bedeutung
Textfelder:	
C$(0:255)	Druckzeichentabelle
LN$(0:8)	Speicher für Druckerausgabe
Z$(0:2,0:4,0:MM)	interner Vokabelspeicher
Numerische Felder:	
K%(0:MM)	Nr. der noch abzufrag. Vokabeln (0..(NN-1))
L%(0:MM)	Schalter, ob Vokabel aktiv (0=nein, —1=ja)
Y%(0:39)	gerade bearbeitete Zeile als ASCII-Array
Textvariablen:	
A$	Antwort, Hilfszeichenkette, Dateiname bei Löschen
B$	Antwort, Hilfszeichenkette
BA$	40 Leerzeichen
BL$	1 Leerzeichen
CL$	Bildschirm löschen
CS$	Cursor
DN$	Dateiname RUSS.VOK.xxx
DS$	Dateiname auf Datei
EM$	Fehlermeldung von Floppy
QU$	Anführungszeichen
R1$	RVS ON
R2$	RVS OFF
TR$	Trennzeile im Ausdruck zwischen Vokabeln
X$	aktuelle Zeile
Numerische Variablen:	
A	Auswahlcode Hauptmenü, ASCII-Code A$
B	Schalter Abfragerichtung 1=R-D, 2=D-R
EN	Fehlernummer Floppy, Fehlerschalter
I	Laufvariable, Hilfsvariable
IB	lfd. Block: 0=Russisch, 1=Deutsch, 2=Bemerkung, 3=Beispiel Russisch, 4=Beispiel Deutsch
IS	lfd. Spalte innerhalb der Eingabezeile (0..39)
IV	lfd. Vokabel (0..MM)
IY	lfd. Bildschirmzeile
IZ	lfd. Zeile innerhalb Block (0..2)
J	Laufvariable, Hilfsvariable
K	Laufvariable, Hilfsvariable
MM	max. Anzahl Vokabeln minus 1 (Dimension)
NN	Anzahl noch nicht gewußte/geprüfte Vokabeln
ST	Status externer Einheiten
VV	nächste Vokabel im Bereich (0..(NN-1))
ZE	lfd. Zeilennummer im Vokabelausdruck

Variablen im Programm »RUSSVOK«

Hauptprogramm-Teile:

10 - 309	Verlegen des Bildschirm-Zeichensatzes von ROM in RAM durch den Befehl »MEM«. Definition der kyrill. Schriftzeichen für die Bildschirmanzeige durch »DESIGN« und »@xxxxxxxx« und für den Drucker 1525 durch Zeichendefinition im Grafik-Modus (im FeldC$(0:255))
410 - 472	Initialisierung, Programmtitelbild
479 - 630	Anzeige des Auswahlmenüs, Eingabe der Auswahl
640 - 730	Datei-Nr.-Eingabe, wenn gemäß der Auswahl im Menü erforderlich. Datei laden, wenn neu. Verzweigen zur gewünschten Aktion.
805 - 844	Löschen einer Vokabeldatei (Auswahl 6)
905 - 1140	Eingeben und Ändern einer Vokabeldatei (Auswahl 5)
1906 - 2111	Abfragen von Vokabeln (Auswahl 1, 2 oder 3)
2906 - 3030	Ausdrucken einer Vokabeldatei (Auswahl 4)

Unterprogramme:

10010 - 10990	Datei von Diskette laden, nicht belegten Speicher löschen
11010 - 11099	Anzeige Vokabel-Eingabe-/Abfrage-Bild
12020 - 12110	Datei auf Diskette speichern
13000 - 13020	Vakabel »IV« im Vokabelbild komplett zeigen
14000	Warten auf ein Zeichen, das in A$ zur Verfügung gestellt wird
15000 - 15005	Druckzeilen zählen. Wenn Seite voll: Seitenwechsel durch Einfügen von acht Leerzeilen

Beschreibung wichtiger Programmzeilen und Unterprogramme im Programm »RUSSVOK«

Buchstabe	Taste(n)	Buchstabe	Taste(n)
А	A	Р	P
Б	Shift B	С	C
В	B	Т	T
Г	Shift G	У	Y
Д	Shift D	Ф	Shift F
Е	E	Х	X
Ё	Shift O	Ц	Shift Z
Ж	Shift S	Ч	Shift T
З	3 (drei)	Ш	Shift @
И	Shift I	Щ	Shift *
Й	Shift J	Ъ	Shift H
К	K	Ы	Shift Y
Л	Shift L	Ь	Shift W
М	M	Э	Shift E
Н	H	Ю	Shift U
О	O	Я	Shift A
П	Shift P		

Tabelle des russischen Alphabets und der Tastenbelegung auf dem Commodore 64

gestartet werden. Durch Drücken irgendeiner Taste gelangt man in das Auswahlmenü. Nach Eingabe des Auswahlcodes (1 bis 8, eine detaillierte Beschreibung erfolgt später) und Drücken von »RETURN« wird, falls bereits Vokabeln im Speicher des Computers sind, gefragt: »DATEI nnn IM SPEICHER, NEU (J/N) ?«. Wenn man die vorhandenen Vokabeln verwenden will, gibt man »N«, sonst »J«, gefolgt von »RETURN«, ein. Wenn noch keine Vokabeln geladen oder eingegeben wurden oder wenn man eine andere Vokabeldatei laden will (Antwort »J«), wird nach der Datei-Nummer (0 bis 999) gefragt. Nach der Eingabe der Nummer und »RETURN« wird die Datei (wenn sie auf der Diskette vorhanden ist) in den Speicher geladen. Vorsicht! Wenn keine Datei mit der eingegebenen Nummer vorhanden ist, wird der Vokabelspeicher im Computer gelöscht.

Die folgenden Funktionen können im Auswahlmenü gewählt werden:

Abfragen von Vokabeln

Folgende Möglichkeiten gibt es: Abfragen Russisch-Deutsch, Abfragen Deutsch-Russisch und Abfragen mit zufälliger Abfragerichtung. Das zu übersetzende Wort und das Beispiel werden angezeigt, wobei in den Blöcken der Zielsprache Fragezeichen erscheinen. Die Bemerkungen werden immer angezeigt. Man übersetzt nun die angezeigte Vokabel und das Beispiel (im Geist oder auf einem Stück Papier) und drückt dann irgendeine Taste. Daraufhin wird die gesamte Vokabel angezeigt. Man muß nun seine Übersetzung mit der angezeigten vergleichen und entscheiden, ob die Vokabel gewußt (Eingabe »R«) oder nicht gewußt (»F«) wurde. Außerdem ist das Beenden des Abfragens durch die Eingabe »E« möglich. Das Abfragen geht zufallsmäßig so lange weiter, bis alle Vokabeln »sitzen«.

Ausdruck der Vokabeln (4)

Der gesamte Vokabelbestand im Speicher wird ausgedruckt (links Russisch, rechts Deutsch, sowie die mit Stern markierten Bemerkungen). Der Ausdruck ist auf 72zeiliges Papier abgestimmt.

Eingabe, Ändern, Ergänzen, Speichern, Ansehen von Vokabeln (5)

Zu Beginn wird die erste Vokabel angezeigt und steht zur Änderung oder Eingabe zur Verfügung. Die folgenden Tasten dienen zum Positionieren des Cursors beziehungsweise zum Blättern innerhalb des Vokabelspeichers:

— RETURN/SHIFT-RETURN: Zeilenwechsel vorwärts und rückwärts innerhalb eines Blocks.

— f1/f2 (= Shift-f1): Wechsel in den nächsten oder vorherigen Block (also zum Beispiel Wechsel von Russisch nach Deutsch und so weiter).

— f3/f4 (= Shift-f3): Vokabelwechsel, vorwärts oder rückwärts.

— f5/f6 (= Shift-f5): Vokabelwechsel, je fünf Vokabeln vorwärts oder rückwärts.

Wenn sich in einer Zeile bereits Text befindet, steht der Cursor immer am Ende des Textes. Durch Drücken von »DEL« kann man Text löschen. Die deutsche Eingabe erfolgt wie gewohnt (nur Großbuchstaben); die russische Eingabe erfolgt gemäß der beigefügten Tabelle (die auf der Tastatur fehlenden kyrillischen Zeichen sind über Shift-Taste einzugeben).

Nach dem Eingeben beziehungsweise Ändern drückt man zum Abspeichern der Vokabeln auf Diskette die Taste »f7«. Eine eventuell vorhandene Vokabeldatei mit derselben Nummer wird dabei überschrieben. Soll keine Abspeicherung erfolgen, so kommt man durch »f8 (= Shift-f7)« wieder in das Auswahlmenü.

Löschen einer Vokabeldatei (6)

Mit dieser Option kann man eine Vokabeldatei von der Diskette entfernen.

Liste der Vokabeldateien anzeigen (7)

Es wird eine Liste aller Dateien auf der Diskette angezeigt, die »RUSS.VOK.nnn« heißen.

Programmende (8)

Diese Option schließlich beendet das Vokabellernprogramm »RUSSVOK — Russische Vokabeln«.

(Hans Peter Postel)

Dieses Programm ist auf Kassette im Rahmen des Leserservice (s.S.75) unter der Bestellnummer CB 011 erhältlich.

130-200	Vorbereitung und Spielablauf
210-280	Erstellen der Sprites durch DATAS
340	Einlesen der Maschinenroutine
350-470	Einlesen der Scheibenfolgen und ihrer Werte
510-540	Gewinnüberprüfung bei Krone in der Mitte
550-590	Überprüfung der Gewinnstufe und des Starters
610-690	Überprüfung bei keiner Krone in der Mitte
700-730	Risikovariablen festlegen
750-780	Gewinnstufenabfrage
790-880	Abfrage des Gewinns auf der rechten Risikoleiter
900-990	Abfragen des Gewinns auf der linken Risikoleiter
1020-1130	Tastaturabfrage: doppelt oder nichts
1150	Geldgewinn bei Auswahl = 1.20
1170	Zufallswert für die Häufigkeit der einzelnen Gewinnchancen
1180-1450	Blinken der einzelnen Sonderspielwerte
1460-1510	Tastaturabfrage für Sonderspiele
1520-1700	Farben für Gewinnstufe und Starter
1720-1840	Ausspielen der ersten Scheibe
1850-1980	Ausspielen der zweiten Scheibe
1990-2080	Ausspielen dritte Scheibe
2200-2350	Löschen der drei Scheiben
2420-2430	Kapital muß größer als 30 Pfennig sein
2440-2550	Errechnen und Anzeigen des Kapitals und der Sonderspiele
2560-2820	Bildschirmaufbau
2830-3710	Neue Zeichen definieren
3720-4140	Restliche DATAS

Programmbeschreibung »Crown No. 1«

CROWN NO.1

Der C 64 als Spielau[tomat]

Dieses Programm wurde für alle diejenigen geschrieben, die sich auch einmal an einem Geldspielautomaten versuchen wollen, ohne großes Kapital zu investieren.

Crown No. I simuliert den gleichnamigen Automaten in den Gasthäusern und Spielhallen. Durch Sprites und selbstdefinierte Zeichen wurde eine entsprechende Grafik erzielt.

Wichtig!! Vor dem Einladen muß der Speicher hochgePOKEd werden:

POKE 44,64 : POKE 16384,0 : NEW

Technische Daten:

Das Programm selber benötigt 15068 Byte im Speicher

Die Dimensionierungen verbrauchten 811 Byte

Crown No. 1 enthält 5 Sprites

Die Maschinenroutine liegt bei Adresse 49152

Es wurden 79 Zeichen umdefiniert.

Spielanleitung

Crown No. 1 fragt Sie nach dem Start, wieviel Kapital Sie investieren möchten. Sie können bis zu 30 Mark einsetzen. Danach müssen sie eine Weile warten, die Zeichen werden umdefiniert.

Sinn des Spiels ist es, drei gleiche Symbole in den fünf Fenstern zu bekommen. Ist dies erreicht, so gibt es mehrere Möglichkeiten. Drei Kronen ergeben Sonderspiele. Bei Sonderspielen wird jeder Gewinn auf drei Mark erhöht, jedes goldene Feld in der Mitte wird als Gewinn (drei Mark) gewertet. Die Anzahl der Sonderspiele wird durch Drücken einer Taste bei laufender Auswahl bestimmt. Alles wird optisch dargestellt. Bei leuchtendem Starter kann die Auswahl während einer bestimmten Zeit nachgestartet werden. Der Starter erlischt dann. Wenn dreimal der gleiche Geldbetrag in den Fenstern erscheint, dann ist der Gewinn gleich dem angegebenen Wert. Die Krone im mittleren Fenster gilt als Joker für jeden Betrag. Eine Krone allein gewinnt 30 Pfennig. Jeder Gewinn kann durch Drücken der Risikotaste (eine beliebige Taste) verdoppelt oder genullt werden. Dies wird optisch an den äußeren beiden Leitern dargestellt.

Zwei gleiche Wappen erhöhen die dazugehörige Gewinnstufe um eins. Es können bei jeder Risikoleiter höchstfalls zwei Gewinnstufen zustandekommen. Eine Gewinnstufe verdoppelt einen Gewinn auf der Risikoleiter automatisch. Bei drei gleichen Wappen wird der Starter eingeschaltet. Sie können die erste Scheibe nachstarten, wenn das Wort »Start« leuchtet. Bei leuchtendem »Stop« können alle Scheiben gestoppt werden. Kapital und Sonderspiele werden in den oberen Fenstern angezeigt. (C. Vigelius)

Bestellnummer CB009

Space

Das Programm ist so konstruiert, daß selbst ein unerfahrener Computerbesitzer Variationen des Spielablaufs durch einfaches Ändern der Variablen vornehmen kann.

Diese Space-Invader-Version ist den gleichnamigen Telespielen nachempfunden, und zeichnet sich durch die Möglichkeit,

SY	Startadresse für die Maschinenroutine
GE	Momentanes Kapital
VH	Basisadresse der Sprites
T	Variable für Schleifen
AZ%	Reihenfolge der Symbole für die beiden äußeren Walzen
AW	Werte der beiden äußeren Walzen
BZ%	Reihenfolge der Symbole für die mittlere Walze
BW	Werte der innenren Walze
SO	Anzahl der Sonderspiele
SD	Starter ein oder aus
GW	Gewinnwert
GR	Wert für erste Gewinnstufe
GG	Wert für zweite Gewinnstufe
AU	Wert für linke oder rechte Risikoleiter
R1$	String des momentanen Gewinns
R3$	String verdoppelter Gewinn
P3	Wert verdoppelter Gewinn
ZO	Anzahl der Sonderspiele bei Risiko
II	Zufallswert
EP	Tastaturpeek
SE	Farbe Starter
LP	Farbe mittleres Fenster
V1	Wert linkes oberes Fenster
V2	Wert linkes unteres Fenster
W1/2	Gleich V2/V1 rechte Fenster
M1	Wert mittleres Fenster

Variablenliste »Crown No. 1«

ablauf kann man ohne weiteres vornehmen. Zu diesem Zweck unterbricht man das Programm und stellt dann die Variablen entsprechend ein. Das Programm wird dann durch »GOTO 10« fortgesetzt.

Bei Unterbrechung mit der »RUNSTOP/RESTORE«-Taste kann man den Zeichengenerator mit »POKE 53272, 29« wieder einschalten.

Nachdem das Einladen beendet ist, erfolgt ein Rücksprung aus dem Unterprogramm und die Register werden gestellt.

In Zeile 80 erfolgt der Aufruf des Maschinen-Programmes und die Auswertung. Sind alle Alien der ersten Zone vernichtet, so kommt man in die zweite Hälfte des Basic-Programmes, die im Prinzip der ersten Hälfte entspricht.

Das Programm ist auf den Joystick an Port 1 abgestimmt, kann aber auch über »CONTROL«, »2« und »SPACE« gesteuert werden. Dies trifft übrigens für alle Spiele mit Joystick an Port 1 zu.

(Gunther Knöpfle)

Bestellnummer CB010

Invaders

...tomat

leicht anpaßbar zu sein, aus.

Außerdem sind natürlich jede Menge Extras, zum Beispiel sichtbeeinflussende Wolken, ein intelligenter Außerirdischer und ein UFO, das Alien aussetzt, vorhanden.

Nach dem Kampf erwartet der Spieler dann eine Highscore-Tabelle. Das Programm läuft auf dem Commodore 64.

Nachdem das Programm mit »RUN« gestartet wurde, springt der Basic-Interpreter nach Zeile 250, wo er die Maschinenprogramme einliest, und den Zeichengenerator in das Basic-RAM verlegt. Hierzu muß gesagt werden, daß man nach dem ersten »RUN« keine Veränderungen am Basic-Programm mehr vornehmen sollte.

Der Interpreter würde die Programmzeilen neu ordnen und dabei den Zeichengenerator durcheinanderbringen. Änderungen am Spiel-

SYS 30000	Eventuell Invaders bewegen, Joystickabfrage und so weiter. Kommt zurück, wenn entweder keine Invaders mehr vorhanden sind oder wenn die Basis vernichtet ist.
SYS 29995	Wie oben, jedoch ohne Initialisierung am Anfang (kein Reset des Punktstandes etc.)
SYS 31000	Farbe setzen. Obere Hälfte Türkis, untere Hälfte Rot.
SYS 33131	Aufruf der Highscore-Tabelle
SYS 33024, S, Z, A$	Druckt A$ ab Spalte S der Zeile Z aus
SYS 33050, X, Z, C	Zieht auf dem Bildschirm eine vertikale Linie über die Position X mit dem Zeichen Z und der Farbe C.

Die wichtigsten Sys-Aufrufe

Register:	Variable:
20182	NA : Nachladegeschwindigkeit der Basis
20183	MO : Wenn 0, dann keine Unterbrechung am Bildschirmrand
20184	AB : Wenn 1, dann Explosion der Invaders bei Abwehrberührung
20185	FI : Feuergeschwindigkeit der Invaders
20186	IN : Geschwindigkeit der Invaders
20187	ET : Feuergeschwindigkeit des unteren Alien
20188	DE : Verzögerung, nur in Verbindung mit »VE« wirksam (DE*VE)
20189	VE : Wenn 0, dann ausgeschaltet (Beschreibung siehe »DE«)
52802	Wenn 0, dann Wolke 1 eingeschaltet
52804	Wenn 0, dann Wolke 2 eingeschaltet
52806	Wenn 0, dann Ufo ausgeschaltet
	ZB : Punktestand
	ZH : Anzahl zerstörter Sektoren
	ZR: Hilfszeiger für steigenden Schwierigkeitsgrad
	M, J, I, G, T : FOR...NEXT-Variablen
	A, B : Für Auslesen der DATAS
	A$: Eingabe, Auswertung
	SU : Prüfsumme
	M1, M2 : Zeilenzeiger für falsche DATAS

Bei Geschwindigkeiten: Je kleiner der Wert, desto größer die Geschwindigkeit

Variablenliste von »Space Invaders«

Vollautomatisches



Sie kennen sicher auch das leidige Problem: Der Urlaub steht vor der Tür und keiner der Nachbarn will die Blumen versorgen, da sie just in der gleichen Zeit Urlaubsgefühle verspüren. Der Computer schafft Abhilfe.

Klaus Michael gönnt seinem C 64 keinen Urlaub



User Port

Pin	Signal	Bemerkung
1	GND	
2	+5V	MAX. 100 mA
3	RESET	
4	CNT1	
5	SP1	
6	CNT2	
7	SP2	
8	PC2	
9	SER. ATN IN	
10	9 VAC	MAX. 100 mA
11	9 VAC	MAX. 100 mA
12	GND	
A	GND	
B	FLAG2	
C	PB0	
D	PB1	
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	
N	GND	

1 (2) 3 4 5 6 7 8 9 10 11 12

(A) B (C) D E F H J K L M N

Bild 2. Pinbelegung des User-Ports. Benötigt wurden nur die Anschlüsse 2, A und C.

Bild 4. Dieser Stecker ist im Elektronikfachversand oder -handel erhältlich.

```
100 REM ***** UHR STELLEN *****
110 C=56328:REM BASISADRESSE  DER UHR IN CIA 1
120 POKEC+7,PEEK(C+7)AND 127
130 POKEC+6,PEEK(C+6)OR 128
140 INPUT"ZEIT IM FORMAT HHMMSS EINGEBEN";A$
150 IF LEN(A$)<>6 THEN 140
160 H=VAL(LEFT$(A$,2))
170 M=VAL(MID$(A$,3,2))
180 S=VAL(RIGHT$(A$,2))
190 IF H>23 THEN 140
200 IF H>11 THEN H=H+68
210 POKEC+3,16*INT(H/10)+H-INT(H/10)*10
```

Das Listing zu Blumengießen **(Fortsetzung Seite 84)**

Die Idee war, mit einer kleinen Tauchpumpe (Campingartikelbedarf) und einem verzweigten Schlauchsystem die Blumen täglich mit dem kostbaren Naß zu versorgen (Bild 1). Diese kleinen Pumpen haben eine Förderleistung von etwa 14

ießen

l/min., was für die minimalen Ein- und Ausschaltzeiten mechanischer Schaltuhren von 30 Minuten viel zu viel ist. Elektronische Schaltuhren, die individuelle Schaltzyklen zulassen, waren wiederum viel zu teuer.

Die Idee, meinen 64er zum Blumengießen herzunehmen, reifte. Denn schließlich besitzt er einen User-Port, der für solche Aufgaben wie geschaffen ist.

Der User-Port, eine Verbindung zur Außenwelt

Der User-Port ist eine Schnittstelle, mit der der Anwender auf einfache Art und Weise Kontakt zwischen dem Commodore und der Außenwelt herstellen kann. Mit ihm ist es unter anderem möglich, Geräte an- und auszuschalten. Da der User-Port eine 8 Bit breite Datenleitung besitzt (PB0-PB7, siehe Bild 2), können ohne weiteres Klimmzüge bis zu 8 Verbraucher geschaltet werden. Dazu ist jedoch etwas zusätzliche Hardware nötig. Um einen Stromverbraucher zu steuern (in unserem Fall eine kleine Pumpe), benötigt man ein Schaltinterface. Denn die Pumpe braucht mehr Strom als der User-Port zur Verfügung stellt. In meinem Fall baute ich mir ein Interface, das direkt an das normale 220-Volt-Netz angeschlossen werden kann. Mit ihm lassen sich grundsätzlich alle Elektrogeräte schalten. Den Aufbau dieses einfachen Schaltinterfaces zeigt Bild 3. Zusätzlich benötigt man noch eine Verbindung vom Interface zum User-Port. Ein geeigneter Stecker zum Anschluß an den User-Port ist der Cardcon-Stecker von TRW (Nr. 251-12-50-170) (Bild 4).

Zur Absicherung des User-Ports wurde zwischen der eigentlichen Schaltung und dem Port ein 6fach-Inverter eingefügt.

Bild 1. Das Interface steht links neben dem Commodore. Die Pumpe liegt im Eimer, der mit Wasser gefüllt ist. Durch die unterschiedlichen Schlauchdurchmesser kann jede Blume individuell mit der für sie notwendigen Wassermenge beliefert werden.

Bauteile	Bezeichnung
Tauchpumpe und dazu passende Schläuche	Campingartikel
Stecker zum Anschluß an den User-Port	Cardcon-Stecker der Firma TRW (Nr. 251-12-50-170)
elektronische Bauteile:	
IC	6fach-Inverter ITT 7404
R1	Widerstand 1 kOhm
T1	Transistor BC 337
D1	Diode 1N 4001
Rel	Relais 12 V
C1	Kondensator 470 mikro Farad
Gleichrichter	8027
Kosten:	
Tauchpumpe und Schläuche	zirka 20 Mark
Interface und Stecker	zirka 50 Mark

Tabelle. Das brauchen Sie, wenn Sie Ihre Blumen mit dem Commodore gießen wollen.

Blumengießen

Das Programm: eine Uhr

Das Gießprogramm besteht im wesentlichen aus der Programmierung der Echtzeituhr im CIA 1. Dieses Programm entstammt dem Data Becker-Buch 64-intern.

In Zeile 100 bis 270 wird die Uhr gestellt, in Zeile 280 bis 480 die laufende Uhr programmiert. In Zeile 440 wird abgefragt, ob der Zeitpunkt zum Gießen erreicht ist. Wenn dies der Fall ist, wird in die Subroutine 1000 verzweigt. In diesem Unterprogramm wird das Datenrichtungsregister auf Ausgang geschaltet und mit POKE 56577,1 die PB0-Leitung auf High gelegt. In Zeile 1030 schließlich wartet das Programm 20 Sekunden und schaltet in Zeile 1050 die Pumpe wieder aus. Die Warteschleife in Zeile 1030 kann dem jeweiligen Anwendungsfall angepaßt werden.

Das Programm läßt sich natürlich für beliebige Schaltzyklen und Schaltzeiten abändern.

(Klaus Michael)

```
220 IF M>59 THEN 140
230 POKEC+2,16*INT(M/10)+M-INT(M/10)*10
240 IFS>59 THEN 140
250 POKEC+1,16*INT(S/10)+S-INT(S/10)*10
260 POKEC,0
270 PRINT""
280 REM ***** UHR LAEUFT *****
290 H=PEEK(C+3):M=PEEK(C+2):S=PEEK(C+1):T=PEEK(C)
300 FL=1
310 IF H>32 THEN H=H-128:FL=0
320 H=INT(H/16)*10+H-INT(H/16)*16:ON FL GOTO 350
330 IF H=12 THEN 360
340 H=H+12
350 IF H=12 THEN H=0
360 M=INT(M/16)*10+M-INT(M/16)*16
370 S=INT(S/16)*10+S-INT(S/16)*16
380 T$=STR$(T)
390 H$=STR$(H):IF LEN(H$)=2 THEN H$=" 0"+RIGHT$(H$,1)
400 M$=STR$(M):IF LEN(M$)=2 THEN M$=" 0"+RIGHT$(M$,1)
410 S$=STR$(S):IF LEN(S$)=2 THEN S$=" 0"+RIGHT$(S$,1)
420 GI$=RIGHT$(H$,2)+RIGHT$(M$,2)+RIGHT$(S$,2)
430 REM **** ZEITPUNKT ZUM GIESSEN ERREICHT? ****
440 IFVAL(GI$)=205500THENGOSUB1000
450 PRINT"";
460 PRINT RIGHT$(H$,2)":"RIGHT$(M$,2)":"RIGHT$(S$,2)":0";
470 PRINT RIGHT$(T$,1)
480 GOTO 290
1000 REM ***** SCHALTUNTERPROGRAMM *****
1010 REM **** PUMPE EINSCHALTEN ****
1020 POKE56579,1:POKE56577,1
1030 FORI=1TO17500:NEXT:REM PUMPE 20 SEC EINGESCHALTET
1040 REM **** PUMPE AUSSCHALTEN ****
1050 POKE56577,0
1060 RETURN

READY.
```

Das Listing zum Blumengießen (Schluß)

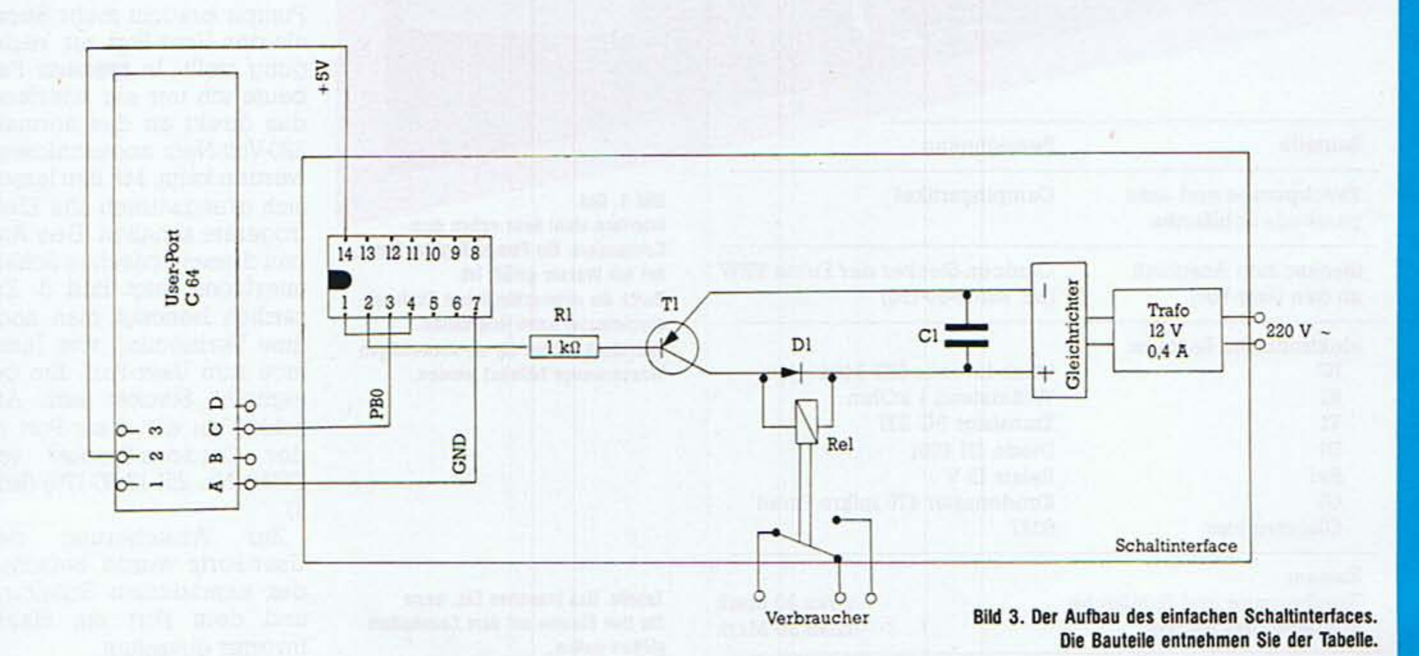


Bild 3. Der Aufbau des einfachen Schaltinterfaces. Die Bauteile entnehmen Sie der Tabelle.

Das erste »Strubs«-Listing

Unser Kurs über den Precompiler »Strubs« ist erst in dieser Ausgabe beendet worden. Und doch hat uns schon jetzt das erste mit »Strubs« erstellte Listing erreicht.

Ich war so begeistert, daß ich mich gleich hingesetzt und ein Programm geschrieben hatte, um die Fähigkeiten von Strubs zu testen, und ich muß sagen, Strubs ist ein Programm, das hält, was es verspricht.

Das von mir mit Strubs entwickelte Programm »Menü« liegt als Quell- und Objektprogramm vor. Der Benutzer kann damit seine Programme nur mit Hilfe der Funktionstasten und der Returntaste auswählen und starten, so daß die umständliche Ladeprozedur per Hand entfällt.

Will man mit »Menü« arbeiten, trägt man zunächst in Data-Zeilen ab Zeilennummer 10 den Text, der auf dem Bildschirm erscheinen soll (maximal 34 Zeichen ein und dann in der gleichen Zeile — durch Komma getrennt — den genauen Namen des Programms, das geladen werden soll. Die Verwendung des Jokers (*) zur Abkürzung ist dabei auch möglich. Anschließend speichert man Menü ab und kann es starten.

»Menü« faßt immer 10 Programme auf einer Bildschirmseite zusammen und blättert — je nachdem man mit dem Pfeil, der auf die Programme zeigt, nach unten oder nach oben aus dem Bildschirm heraus will — weiter oder zurück.

Bei Programmen, die größer sind als »Menü« ist darauf zu achten, daß in der ersten bearbeiteten Programmzeile eine Routine steht, die die Pointer auf das Basic-Ende ausrichtet (siehe dazu Zeile 1 von »Menü«).

Ersetzt man in den Programmen, die durch »Menü« geladen werden sollen noch die END-Anweisung — oder das Programmende, falls keine END vorhanden — durch LOAD"MENUE",8, so wird nach Beendigung des gewählten Programms automatisch wieder »Menü« geladen.

Vorsicht ist geboten bei Programmen, die den BASIC-Anfang oder das RAM-Ende verschieben, da diese Programme zum Systemabsturz führen, wenn man die entsprechenden Vektoren nicht zurücksetzt.

»Menü« müßte auch auf dem C 64 laufen, ich konnte es aber leider nicht testen. (Gerd Sombetzki)

Folgende Änderungen nahm ich für meinen VC 20 mit 40/80 Zeichenkarte, 32 K-RAM, Floppy 1541 und VC 1526 vor. Zunächst verlegte ich die Maschinenroutine in den Kassettenpuffer. Zeile: 45610 POKE 828 + i, w: ...usw.
Der Sprung zur Interpreterroutine \$A7E7 = #167 liegt beim VC 20 bei \$C7E7 = #199. Außerdem fehlte das im Assemblerlisting vorhandene PLP = \$28 = #40.

```
D.h.: Zeile 45620 DATA32, 115,0,8,201,33,240,4,40,76,231,
            199,40
            45630 DATA169,18,133,44,169,138,76,231,199,999
Verschieben der Initialisierungsroutine:
            45650 FORI=0TO10:READW:POKE850+I,W
            45670 SYS850
            45680 DATA169,60,141,8,3,169,3,141,9,3,96
```

Die Änderung des Wertes 8 in 18 in der Zeile 45630 ist notwendig, da der Basic-Anfang beim VC-20 mit Erweiterung) = 8K bei \$1200 beginnt, also Highbyte \$12 = #18.

```
0 '========= DIE ZEILE 1 RICHTET DIE POINTER AUF DAS BASIC-ENDE AUS. ========
1 POKE 45,PEEK(174) : POKE 46, PEEK(175) : CLR : GOTO £HPTPRG
2 '.......... AB ZEILE 10 STEHEN DIE MENUEDATEN ...............
5 REM"DATAZEILEN_ENTHALTEN_<PROGRAMMNAME>,_<LADENAME>
98 DATA "ENDE............................", "..................."
100 DATA @,@
200 '
1000 £MENUE
1030 :        !IF AN/10-INT(AN/10) = 0 THEN
1040 :           M'ASKEN'D'URCHLAEUFE' = AN/10-1
1050 :        !ELSE
1060 :           MD = INT(AN/10)
1070 :        !FI
1080 '.......HAUPTSCHLEIFE........
1090 :        M'ASKENZAEHLER' = 0
1100 :        !LOOP
1110 :           GOSUB £MASKE
1120 :           T'ASTENZAEHLER' = 1
1130 :           !LOOP
1140 :              GOSUB £PFEIL : GOSUB £TASTE
1150 :              !CASEOF TA'STE' = 0  THEN
1160 :                    !EXIT
1170 :                 !OF TA'STE' = -1 THEN
1180 :                    T = T-1 : IF T < 1 THEN !EXIT
1190 :                 !OF TA'STE' =  1 THEN
1200 :                    T = T+1 : IF T > 10 OR T+M*10 > AN THEN !EXIT
1230 :              !ECASE
1240 :           !ELOOP
1250 :           !CASEOF TA'STE' =  0 THEN
1260 :                 !EXIT
1270 :              !OF TA'STE' = -1 THEN
1280 :                 M = M-1 : IF M < 0  THEN M = 0
1290 :              !OF TA'STE' =  1 THEN
1300 :                 M = M+1 : IF M > MD THEN M = MD
1310 :           !ECASE
1320 :        !ELOOP
1325 '.......HAUPTSCHLEIFE ENDE ..
1330 GOTO £LADEN
1340 '
1500 £MASKE
1510 :           GOSUB £ZEILE1
1520 :           I = M*10+1 ' M IST MASKENSEITENZAEHLER
1530 :           !WHILE I <= M*10+10 AND I <= AN'ZAHL' !DO
1540 :              PRINT : PRINT TAB(5) P'RG'N'AME'$(I)
1550 :              I = I+1
1560 :           !EWHILE
1570 :           GOSUB £HELP
1580 RETURN
1590 '
1700 £ZEILE1
1710 :  PRINT CHR$(147)TAB(12)"PROGRAMMAUSWAHL"
1720 :  PRINT              TAB(12)"==============="
1730 RETURN
1740 '
1800 £HELP
1810 :           Z = 24 : S = 2 : GOSUB £CRSRPO.
1820 :           PRINT RO$"F1 => OBEN"RF$" ";
1823 :           PRINT RO$"F3 => UNTEN"RF$" ";
1826 :           PRINT RO$"RETURN => LOAD"RF$;
1830 RETURN
1840 '
1900 £CRSRPO.
1910 :           PRINT CHR$(19) LEFT$(CD$,Z-1) LEFT$(CR$,S-1);
1920 RETURN
1930 '
2000 £PFEIL
2005 '
2010 '         Q IST DER VEKTOR ZUM LOESCHEN ODER SETZEN DES PFEILS
2020 :
2030 :           Z = T*2+2 : S = 2 : GOSUB £CRSRPO.
2040 :           IF Q = 1 THEN £LBL1
2050 : £LBL1   PRINT "==>" : GOTO £LBL2
2060 : £LBL2   PRINT "   " : Q = 0
2070 RETURN
2072 '
2200 £TASTE
2210 :
2220 :           !LOOP
2230 :              GET TA'STE'$ : IF TA'STE'$ = "" THEN £THIS
2240 :              !CASEOF TA'STE'$ = CHR$(133) THEN 'F1-TASTE GEDR.
2250 :                 Q = 1 : GOSUB £PFEIL
2260 :                 TA'STE' = -1 : !EXIT
2270 :              !OF     TA'STE'$ = CHR$( 13) THEN 'RETURN   GEDR.
                        TA'STE' =  0 : !EXIT
```

Strubs-Listing »Menü« (Anfang)

Strubs-Listing »Menü« (Schluß)

```
2280 :
2290 :
2300 :          !OF       TA'STE'$ = CHR$( 134) THEN 'F3-TASTE GEDR.
2310 :                 Q = 1 : GOSUB £PFEIL
2320 :                 TA'STE' = 1 : !EXIT
2330 RETURN     !ECASE
2332 ,       !ELOOP
2400 £LADEN
2410 :
2420 :      NU'MMER' = M'ASKENZAEHLER' * 10 + T'ASTENZAEHLER
2430 :      Z'EILE' = T*2+2 : S'PALTE' = 6 : GOSUB £CRSRPO.
2440 :      PRINT RO$ P'RG'N'AME'$(NU) RF$
2450 ,      IF LEFT$(P'RG'N'AME'$(NU),4) = "ENDE" THEN £SCHLUSS
2460 ,      (NOCH ZU IMPLEMENTIERENDE FUNKTION: LADEN VON MPGS)
2470 :      LOAD L'ADE'N'AME'$(NU),8
2490 'ENDE
2495 ,
2700 £SCHLUSS
2710 :
2720 :      PRINT CHR$( 147);
2730 :      PRINT LEFT$(CD$,10) TAB(5);
2740 :      PRINT          RO$ "    F5 => MENUE          "
2750 :      PRINT TAB(5) RO$ "    F7 => ANDERE DISK"
2760 :      PRINT TAB(5) RO$ "RETURN => ENDE           "
2770 :      !LOOP
2780 :        GET TA'STE'$ : IF TA'STE'$ = "" THEN £THIS
2790 :        !CASEOF TA'STE'$ = CHR$( 135) THEN 'F5-TASTE GEDR.
2800 :          RUN10000
2810 :        !OF       TA'STE'$ = CHR$( 136) THEN 'F7-TASTE GEDR.
2820 :          PRINT : PRINT TAB(5) "BITTE ANDERE DISK EINLEGEN,"
2830 :                  PRINT TAB(5) "DANN BEL. TASTE DRUECKEN!"
2840 :          GET TA'STE'$ : IF TA'STE'$ = "" THEN £THIS
2850 :          OPEN15,8,15,"I" : CLOSE15
2860 :          LOAD "MENUE",8
2870 :      !OF       TA'STE'$ = CHR$( 13) THEN 'RETURN   GEDR.
2880 :          PRINT CHR$( 147)
2890 :          PRINT "BITTE DISKETTE DEM LAUFWERK ENTNEHMEN,"
2900 :          PRINT "DANN LAUFWERK UND RECHNER ABSCHALTEN."
2910 :          PRINT : PRINT "BY, BY!"
2920 :          END
2930 :        !ECASE
2940 ,      !ELOOP
10000 £HPTPRG
10010 ,
10012 REM"      ******** HAUPTPROGRAMM ***********
10014 REM"_____MENUE__VERSION__VOM__01.05.84__
10016 REM"_________GERD____SOMBETZKI__________
10018 REM"_________FUHRMANNSTR.___47__________
10020 REM"_________4600_DORTMUND__13__________
10022 ,  _________TEL.:_0231/213656__________
10030 :
10040 :
10050 ,            GOSUB £INIT
10060 ,            GOTO £MENUE
10070 ,
10100 £INIT      ************************************
10130 :
10140 :
10150 :      GOSUB £ANZAHL : IF AN'ZAHL' = 0 THEN £SCHLUSS
10160 :      DIM P'RG'N'AME'$(AN'ZAHL'), L'ADE'N'AME'$(AN'ZAHL')
10170 :      RESTORE
10180 :      FOR I = 1 TO AN'ZAHL
10190 ,        READ P'RG'N'AME'$(I), L'ADE'N'AME'$(I)
10200 ,      NEXT
10210 ,-----------GLOBALVARIABLENDEFINITION-----------------
10220 :
10230 :      CD$ = "■" 'CURSOR DOWN' : RO$ = "■" 'RVS ON
10235 ,      CR$ = "■" 'CURSOR RIGHT' : RF$ = "■" 'RVS OFF
10240 :
10250 :      FOR I = 1 TO 7
10260 :        CD$ = CD$+CD$ : CR$ = CR$+CR$
10290 RETURN NEXT
10295 ,
10300 £ANZAHL
10310 :
10320 :
10330 :      RESTORE : AN'ZAHL' = -1
10340 :      !REPEAT
10350 :        READ P'RG'N'AME'$, L'ADE'N'AME'$
10360 RETURN   AN'ZAHL' = AN'ZAHL' + 1
10370 ,      !UNTIL P'RG'N'AME'$ = "@"

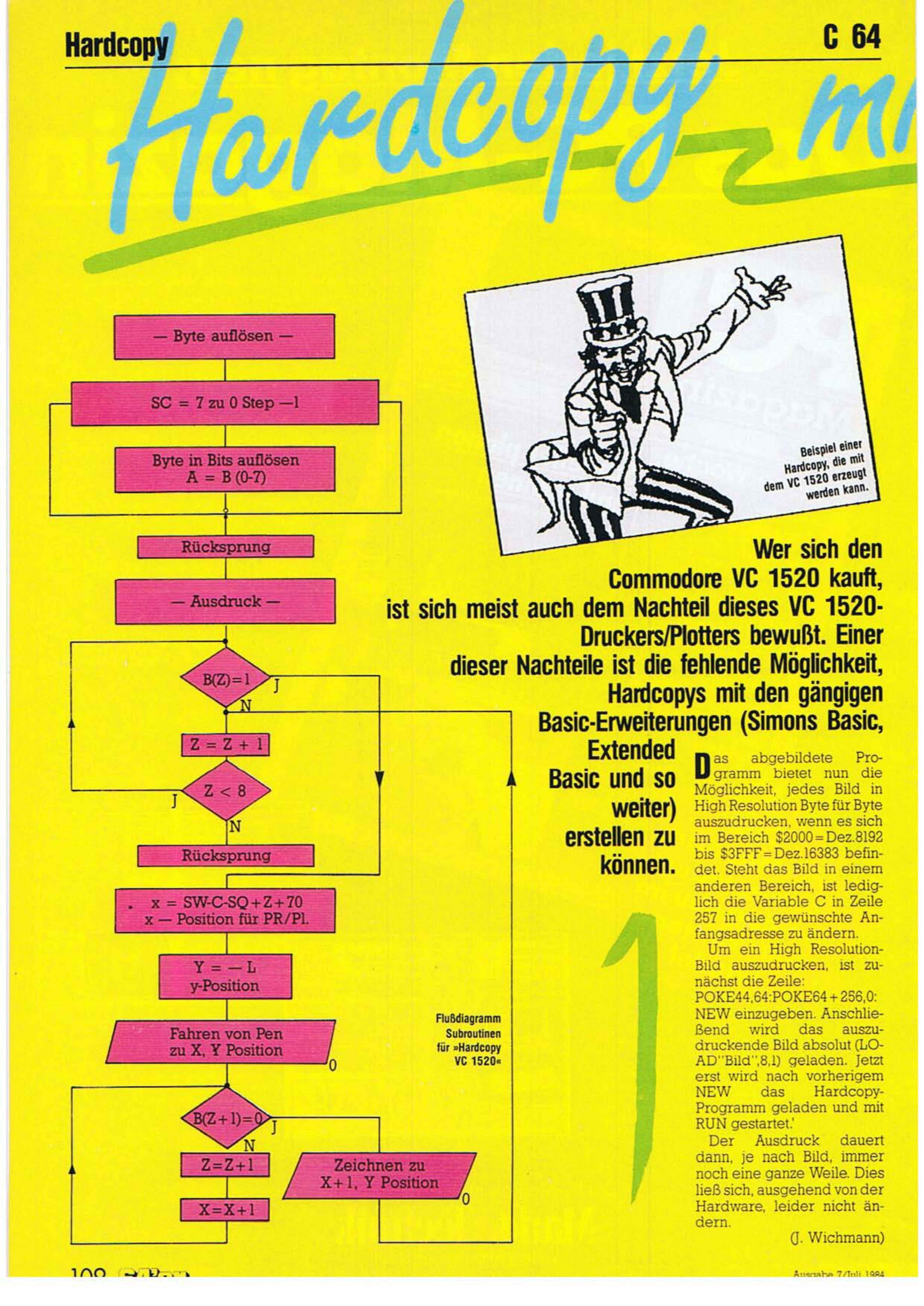
READY.
```

```
1 POKE45,PEEK(174):POKE46,PEEK(175):CLR:GOTO10000
5 REM"DATAZEILEN_ENTHALTEN_<PROGRAMMNAME>,_<LADENAME>
98 DATA"ENDE.........................","...............
100 DATA@,@
1000 :
1030 IFNOT(AN/10-INT(AN/10)=0)THEN1051
1040 MD=AN/10-1
1050 GOTO1070
1051 :
1060 MD=INT(AN/10)
1070 :
1090 M=0
1100 :
1110 GOSUB1500
1120 T=1
1130 :
1140 GOSUB2000:GOSUB2200
1150 IFNOT(TA=0)THEN1171
1160 GOTO1241
1170 GOTO1230
1171 IFNOT(TA=-1)THEN1191
1180 T=T-1:IFT<1THEN1241
1190 GOTO1230
1191 IFNOT(TA=1)THEN1230
1200 T=T+1:IFT>10ORT+M*10>ANTHEN1241
1230 :
1240 GOTO1130
1241 :
1250 IFNOT(TA=0)THEN1271
1260 GOTO1321
1270 GOTO1310
1271 IFNOT(TA=-1)THEN1291
1280 M=M-1:IFM<0THENM=0
1290 GOTO1310
1291 IFNOT(TA=1)THEN1310
1300 M=M+1:IFM>MDTHENM=MD
1310 :
1320 GOTO1100
1321 :
1330 GOTO2400
1500 :
1510 GOSUB1700
1520 I=M*10+1
1530 IFNOT(I<=M*10+10ANDI<=AN)THEN1561
1540 PRINT:PRINTTAB(5)PN$(I)
1550 I=I+1
1560 GOTO1530
1561 :
1570 GOSUB1800
1580 RETURN
1700 :
1710 PRINTCHR$(147)TAB(12)"PROGRAMMAUSWAHL"
1720 PRINTTAB(12)"==============="
1730 RETURN
1800 :
1810 Z=24:S=2:GOSUB1900
1820 PRINTRO$"F1 => OBEN"RF$" ";
1823 PRINTRO$"F3 => UNTEN"RF$" ";
1826 PRINTRO$"RETURN => LOAD"RF$;
1830 RETURN
1900 :
1910 PRINTCHR$(19)LEFT$(CD$,Z-1)LEFT$(CR$,S-1);
1920 RETURN
2000 :
2020 Z=T*2+2:S=2:GOSUB1900
2030 IFQ=1THEN2050
2040 PRINT"==>":GOTO2060
2050 PRINT"   ":Q=0
2060 :
2070 RETURN
2200 :
2210 :
2220 GETTA$:IFTA$=""THEN2220
2230 IFNOT(TA$=CHR$(133))THEN2261
2240 Q=1:GOSUB2000
2250 TA=-1:GOTO2321
2260 GOTO2310
2261 IFNOT(TA$=CHR$(13))THEN2281
2270 TA=0:GOTO2321
2280 GOTO2310
2281 IFNOT(TA$=CHR$(134))THEN2310
2290 Q=1:GOSUB2000
2300 TA=1:GOTO2321
2310 :
2320 GOTO2210
2321 :
2330 RETURN
2400 :
2410 NU=M*10+T
2420 Z=T*2+2:S=6:GOSUB1900
2430 PRINTRO$PN$(NU)RF$
2440 IFLEFT$(PN$(NU),4)="ENDE"THEN2700
2470 LOADLN$(NU),8
2700 :
2710 PRINTCHR$(147);
2720 PRINTLEFT$(CD$,10)TAB(5);
2730 PRINTRO$"     F5 => MENUE      "
2740 PRINTTAB(5)RO$"     F7 => ANDERE DISK"
2750 PRINTTAB(5)RO$"RETURN => ENDE        "
2760 :
2770 GETTA$:IFTA$=""THEN2770
2780 IFNOT(TA$=CHR$(135))THEN2801
2790 RUN10000
2800 GOTO2920
2801 IFNOT(TA$=CHR$(136))THEN2861
2810 PRINT:PRINTTAB(5)"BITTE ANDERE DISK EINLEGEN,
2820 PRINTTAB(5)"DANN BEL. TASTE DRUECKEN!"
2830 GETTA$:IFTA$=""THEN2830
2840 OPEN15,8,15,"I":CLOSE15
2850 LOAD"MENUE",8
2860 GOTO2920
2861 IFNOT(TA$=CHR$(13))THEN2920
2870 PRINTCHR$(147)
2880 PRINT"BITTE DISKETTE DEM LAUFWERK ENTNEHMEN,"
2890 PRINT"DANN LAUFWERK UND RECHNER ABSCHALTEN."
2900 PRINT:PRINT"BY, BY!"
2910 END
2920 :
2930 GOTO2760
2931 :
10000 :
10012 REM"_____MENUE__VERSION__VOM__01.05.84__
10014 REM"____________GERD_____SOMBETZKI____________
10016 REM"____________FUHRMANNSTR.___47_____________
10018 REM"____________4600_DORTMUND__13_____________
10020 REM"____________TEL.:_0231/213656_____________
10030 GOSUB10100
10040 GOTO1000
10100 :
10130 GOSUB10300:IFAN=0THEN2700
10140 DIMPN$(AN),LN$(AN)
10150 RESTORE
10160 FORI=1TOAN
10170 READPN$(I),LN$(I)
10180 NEXT
10220 CD$="■":RO$="■"
10230 CR$="■":RF$="■"
10240 FORI=1TO7
10250 CD$=CD$+CD$:CR$=CR$+CR$
10260 NEXT
10290 RETURN
10300 :
10310 RESTORE:AN=-1
10320 :
10330 READPN$,LN$
10340 AN=AN+1
10350 IFNOT(PN$="@")THEN10320
10360 RETURN

READY.
```

So würde das Programm ohne den Precompiler aussehen.

Hardcopy mit dem VC 1520



Flußdiagramm Subroutinen für »Hardcopy VC 1520«

Beispiel einer Hardcopy, die mit dem VC 1520 erzeugt werden kann.

Wer sich den Commodore VC 1520 kauft, ist sich meist auch dem Nachteil dieses VC 1520-Druckers/Plotters bewußt. Einer dieser Nachteile ist die fehlende Möglichkeit, Hardcopys mit den gängigen Basic-Erweiterungen (Simons Basic, Extended Basic und so weiter) erstellen zu können.

Das abgebildete Programm bietet nun die Möglichkeit, jedes Bild in High Resolution Byte für Byte auszudrucken, wenn es sich im Bereich $2000 = Dez.8192 bis $3FFF = Dez.16383 befindet. Steht das Bild in einem anderen Bereich, ist lediglich die Variable C in Zeile 257 in die gewünschte Anfangsadresse zu ändern.

Um ein High Resolution-Bild auszudrucken, ist zunächst die Zeile:
POKE44,64:POKE64+256,0:
NEW einzugeben. Anschließend wird das auszudruckende Bild absolut (LOAD"Bild",8,1) geladen. Jetzt erst wird nach vorherigem NEW das Hardcopy-Programm geladen und mit RUN gestartet.'

Der Ausdruck dauert dann, je nach Bild, immer noch eine ganze Weile. Dies ließ sich, ausgehend von der Hardware, leider nicht ändern.

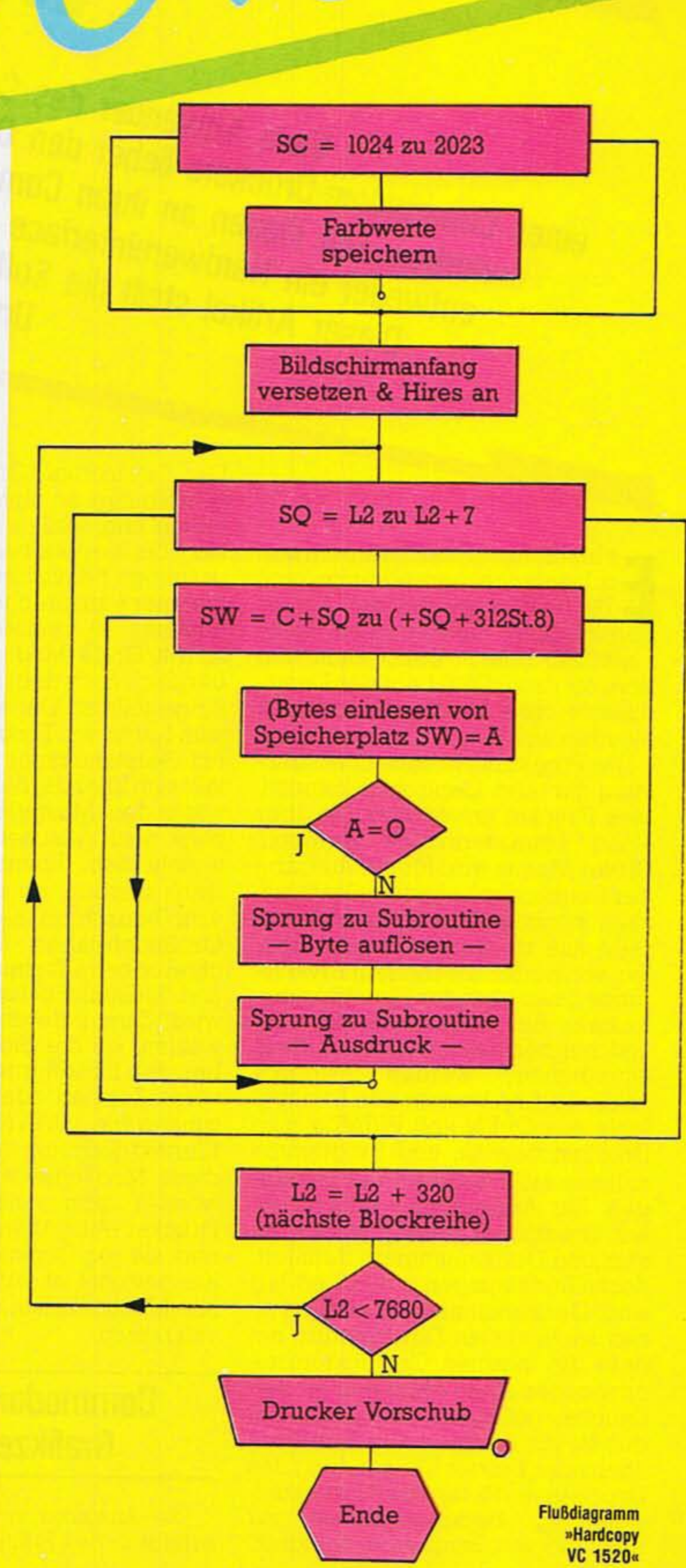
(J. Wichmann)

t dem VC 1520

```
1 REM
2 REM
3 REM* SPRUNG HAUPTPROGR. *
4 REM
5 REM
6 GOTO200
10 REM
20 REM
30 REM*   SUB ROUTINE      *
40 REM*   BYTE AUFLOESEN *
50 REM
60 REM
70 FORSC=7TO0STEP-1
80 :IFA-2^SC<0THENB(7-SC)=0:GOTO100
90 :B(7-SC)=1:A=A-2^SC
100 NEXTSC
110 RETURN
200 REM
210 REM
220 REM* EINLESEN VON BYTES *
230 REM* HAUPT-ALGORITHMUS  *
240 REM
250 REM
256 FORSC=1024TO2023:POKESC,16:NEXT:POKE
53265,59:POKE53272,24
257 C=8192
260 DIMB(9):REM* KONSTANTE *
270 OPEN1,6,1:REM* 1520 AUF X/Y PLOT *
275 FORSQ=L2TOL2+7:L=L+1
280 :FORSW=C+SQTOC+SQ+312STEP8
290 :A=PEEK(SW):IFA=0THEN320
295 :GOSUB70
300 :GOSUB450
320 :Z=0:NEXTSW
325 NEXTSQ
330 L2=L2+320:IFL2<7680THEN275
340 REM
350 REM* ENDE *
360 REM
370 CLOSE1:OPEN4,6:REM* 1520 AUF ASCII *
380 FORSC=1TO3:PRINT#4,""
390 NEXTSC:CLOSE4:END:REM**************
* VORSCHUB DES 1520 *
400 REM
410 REM
420 REM* AUSDRUCKEN AUF PRINTER/PLOTTER
430 REM* 1520 IN HIRES *
440 REM
450 REM
452 IFB(Z)=1THEN460
454 Z=Z+1:IFZ<8THEN452
456 RETURN
460 X=SW-C-SQ+Z+70:Y=-L:PRINT#1,"M",X,Y
510 IFB(Z+1)=0THENPRINT#1,"D",X+1,Y:GOTO
454
520 Z=Z+1:X=X+1:GOTO510

READY.
```

Listing »Hardcopy mit dem VC 1520«



Flußdiagramm »Hardcopy VC 1520«

Komfortables Treiberprogramm für

Viele Anwender des Commodore 64 möchten anstelle eines Commodore-Druckers lieber den Drucker eines anderen Herstellers erwerben. Um diesen an ihren Computer anzuschließen, müssen sie entweder ein Hardwareinterface oder eine Softwarelösung kaufen. Dieser Artikel stellt die Softwarelösung für den Anschluß von Druckern am Userport des C 64 vor.

Es handelt sich hierbei um ein Maschinenspracheprogramm mit 726 Byte Länge, das den Adreßbereich CB57 - CBD1 (dezimal: 51456 bis 52182) belegt. Damit kann zum Beispiel das DOS 5.1 auf der Demodiskette ohne Probleme zusätzlich geladen werden.

Das Programm ist ohne Einschränkung für alle Centronics-kompatiblen Drucker anwendbar, die über einen Bitmustermodus verfügen. Dieser Modus wird für die Ausgabe der Commodore-eigenen Grafikzeichen benötigt. Das Programm besteht aus mehreren Programmteilen, von denen die meisten Erweiterungen bestehender Ein-/Ausgaberoutinen des Betriebssystems sind und bei der Initialisierung in diese eingebunden werden. Dadurch können schon bestehende Basicbefehle wie OPEN und PRINT# zum Drucken benutzt, und Programme müssen nicht umgeschrieben werden. Zur Ansteuerung verschiedener Druckmodi wurden jedoch zusätzliche Gerätenummern definiert, deren Bedeutungen in Bild 1 erklärt sind. Gerätenummer 16 realisiert einen sogenannten Direktmodus, mit dem die internen Commodorezeichencodes ohne Wandlung an den Drucker gelangen. Der Direktmodus ist zur Ausgabe von Steuerzeichen oder bei der Verwendung des Druckers als Plotter zur Einzelnadelsteuerung beziehungsweise zur Ausgabe von Bitmustern geeignet. Der Textmodus (Groß- und Kleinschreibung) ist unter Gerätenummer 18 und, weil er wohl am häufigsten bei bereits bestehenden Programmen benutzt ist, unter Gerätenummer 4 ansprechbar. Die Gerätenummer 19 realisiert den Großschrift/Grafik-Modus, wie er beim C 64 gleich nach dem Einschalten voreingestellt ist. Der wichtige Modus zum Listen von Programmen wurde mit Gerätenummer 17 realisiert. Es läßt sich über die Sekundäradresse, wie in dem Musterlisting (Bild 4) gezeigt, noch zwischen vier Fällen unterscheiden. Einmal kann ausgewählt werden, ob das Listing, wie vom Bildschirm her gewohnt, mit Großbuchstaben und Grafikzeichen oder im Textmodus mit großen und kleinen Buchstaben gedruckt wird. Zum anderen kann man auswählen, ob die Steuerzeichen wie bei der Bildschirmausgabe als inverse Zeichen oder durch Abkürzungen wie ⟨CRD⟩ (Cursor down) im Klartext gedruckt werden. Durch all diese Möglichkeiten kann der Anwender zum einen auf seinem Drucker Ausgaben erzeugen, wie man sie von Commodore-Druckern her gewohnt ist, zum anderen auch alle Möglichkeiten seines Druckers voll nutzen.

Commodore-eigene Grafikzeichen

Die Ausgabe von Grafikzeichen erfolgt in der Routine OUTGEN ab Adresse $CAC4. Zu diesem Zweck wird der Zeichengenerator des C 64 ausgelesen. Die Zeichen, die aus einer 8 x 8 Punktmatrix bestehen, sind dort in je 8 Byte abgelegt. Jedes Byte repräsentiert das Punktmuster einer Zeile der Matrix. Ein Matrixdrucker druckt die Zeichen, indem er sie ebenso als Muster von matrixartig angeordneten Punkten zu Papier bringt. Jedoch gibt er die Punktmatrix nicht wie der Bildschirm zeilenweise sondern spaltenweise aus. Deshalb wird in OUTGEN die im Zeichengenerator in 8 Zeilenbyte gespeicherte Zeichenmatrix zunächst in 8 Spaltenbyte umorganisiert. Nach Umschalten des Druckers in den sogenannten Bitmustermodus, in dem er jedes ankommende Datenbyte nicht mehr als ASCII-Zeichen sondern als Musterbyte für eine Matrixspalte interpretiert, gibt OUTGEN die acht geänderten Zeichenbyte an den Drucker aus. Dieser fügt sie wieder zu einem 8 x 8 Punktmatrixzeichen zusammen. So ist es möglich, auf einem Matrixdrucker exakte Kopien der C 64-Bildschirmzeichen herzustellen, obwohl dieser nicht über den entsprechenden Zeichensatz verfügt.

Handhabung des Programms

Die etwas Konzentration fordernde Methode der Programmeingabe besteht im Eintippen des abgebildeten Basicprogramms (Bild 2), in dem das Treiberprogramm in DATA-Zeilen steht. Ein eingebauter Quersum-



Centronics-Drucker

GERÄTENUMMER 16	=	DIREKTMODUS
GERÄTENUMMER 18,4	=	TEXTMODUS
GERÄTENUMMER 19	=	GRAFIKMODUS
GERÄTENUMMER 17	=	LIST-MODUS

SEK-ADR.	MODUS	STEUERZEICHEN
0	NORMAL	NORMAL
1	KLEIN	NORMAL
2	NORMAL	ERKLÄRT
3	KLEIN	ERKLÄRT

Bild 1. Die Bedeutungen der Gerätenummern

mentest deckt hoffentlich Eingabefehler auf.

Zur Herstellung des Verbindungskabels zwischen Userport und Drucker ist in Bild 3 eine Verbindungstabelle angegeben. Das Kabel sollte für eine störungsfreie Funktion nicht länger als ein Meter lang sein und aus einem abgeschirmten, mehradrigen Steuerkabel bestehen, das man in (fast) jedem Elektronik-Bastelgeschäft findet. Dort sind auch meist der Centronicsstecker und der Stecker für den Userport erhältlich.

Initialisiert wird die geladene Treiberroutine mit
SYS 12*4096 + 9*256 beziehungsweise SYS 51456

Hierbei wird die Routine in das Betriebssystem eingebunden. Jedoch Vorsicht: Nach einem Break, zum Beispiel durch die Betätigung der Tasten RUN/STOP und RESTORE ausgelöst, muß die Routine erneut initialisiert werden, da die I/O-Vektoren vom Betriebssystem zurückgesetzt wurden. Die einzelnen Druckmodi spricht man mit den üblichen Basicbefehlen an. Geöffnet wird der Ausgabekanal mit:
OPEN log. Dateinummer, Geräteadr. [,Sekundäradr.]
Die eckigen Klammern kennzeichnen optionale Angaben. So dann kann auf den geöffneten Kanal mit
PRINT # log. Dateinummer ausgegeben werden. Ein Programmlisting wird zum Beispiel erzeugt mit (Bild 4) den Befehlen
OPEN 17,17 [Sekundäradr.]:CMD17:
LIST
PRINT#17 : CLOSE 17

Der PRINT-Befehl vor dem CLOSE ist notwendig, damit der CMD-Modus aufgehoben wird.

Umstellen auf beliebige Drucker mit Centronics-Schnittstelle

Das Programm wurde für einen Epson-Drucker geschrieben. Unverändert ist es für jeden anderen

Bild 3. Der Verkabelungsplan

USER PORT	— CENTRONICS	
A	GND	16
B	FLAG — BUSY	11
C	D0	2
D	D1	3
E	D2	4
F	D3	5
H	D4	6
J	D5	7
K	D6	8
L	D7	9
M	PA2 — STROBE	1

```
100 FOR I=51456 TO 52182
110 : READ X: POKE I,X: S=S+X
120 NEXT I
130 IF S<>82731 THEN PRINT"FEHLER IN DAT
AS !!": END
140 SYS 51456
150 REM
160 REM CENTRONICS TREIBERROUTINE
170 REM
200 DATA 169, 90,160,201,141, 26,  3,140
201 DATA  27,  3,169,145,160,201,141, 28
202 DATA   3,140, 29,  3,169,173,160,201
203 DATA 141, 30,  3,140, 31,  3,169,200
204 DATA 160,201,141, 32,  3,140, 33,  3
205 DATA 169,227,160,201,141, 38,  3,140
206 DATA  39,  3,169,255,141,  3,221,173
207 DATA   2,221,  9,  4,141,  2,221, 96
208 DATA  72,169, 16, 44, 13,221,240,251
209 DATA 104,141,  1,221,173,  0,221,  9
210 DATA   4,141,  0,221, 41,251,141,  0
211 DATA 221, 96,166,184,240,  5, 32, 15
212 DATA 243,208,  3, 76,254,246,166,152
213 DATA 224, 10,144,  3, 76,251,246,230
214 DATA 152,165,184,157, 89,  2,165,185
```

Bild 2. Basic-Lader für das Treiberprogramm

```
215 DATA   9, 96,157,109,  2,165,186,157
216 DATA  99,  2,201,  4,240,  4,201, 16
217 DATA 144,  2, 24, 96,201,  0, 76,119
218 DATA 243, 32, 20,243,240,  2, 24, 96
219 DATA  32, 31,243,138, 72,165,186,201
220 DATA  16,176,  7,201,  4,240,  3, 76
221 DATA 157,242, 76,241,242, 32, 15,243
222 DATA 240,  3, 76,  1,247, 32, 31,243
223 DATA 165,186,201,  4,240,  4,201, 16
224 DATA 144,  3, 76, 10,247, 76, 25,242
225 DATA  32, 15,243,240,  3, 76,  1,247
226 DATA  32, 31,243,165,186,201,  4,240
227 DATA   4,201, 16,144,  3, 76,117,242
228 DATA  76, 91,242, 72,133,158,165,154
229 DATA 201, 16,176,  7,201,  4,240,  3
230 DATA  76,205,241,152, 72,138, 72,165
231 DATA 158,164,154,192, 16,208,  6, 32
232 DATA  64,201, 24,144, 31,192, 17,208
233 DATA   6, 32, 92,202, 24,144, 21,192
234 DATA   4,240,  4,192, 18,208,  6, 32
235 DATA  43,202, 24,144,  7,192, 19,208
236 DATA   3, 32, 68,202,104,170,104,168
237 DATA 104, 24, 96,201, 65,144, 18,201
238 DATA  95,176,  4,  9, 32,208, 10,201
239 DATA 193,144,  6,201,222,176,  2, 41
240 DATA 127, 76, 64,201,201,255,240, 24
241 DATA 201, 96,176,  3, 76, 64,201,233
242 DATA  64, 16,  2,233, 64,160,208,132
243 DATA   6, 76,196,202,201,255,208,  6
244 DATA 162, 94,160,208,208, 91, 72,164
245 DATA 185,192,255,208,  2,230,185, 41
246 DATA 127,201, 32,144, 44,168,165,185
247 DATA  41,  1,240, 16,104,201,160,144
248 DATA   4,201,192,144,  3, 76, 43,202
249 DATA 233, 64,208, 14,104,201, 96,176
250 DATA   3, 76, 64,201,233, 64, 16,  2
251 DATA 233, 64,160,208,132,  6, 76,196
252 DATA 202,104, 36, 15, 48,  3, 76, 64
253 DATA 201, 24,105, 64, 48,  2,105, 64
254 DATA 170,165,185, 41,  2,208,113,160
255 DATA 208,165,185, 41,  1,240,  2,160
256 DATA 216,132,  6,138,168,169,  0,162
257 DATA   7.157.192.  2.202. 16.250.152
258 DATA  74, 74, 74, 74, 74, 24,101,  6
259 DATA 133,  6,152, 10, 10, 10,133,  5
260 DATA 169,  1,133,  3,120,165,  1, 41
261 DATA 251,133,  1,160,  7,177,  5,133
262 DATA   2,162,  7,  6,  2,144,  8,189
263 DATA 192,  2,  5,  3,157,192,  2,202
264 DATA  16,241,  6,  3,136, 16,230,165
265 DATA   1,  9,  4,133,  1, 88,162, . 0
266 DATA 189,210,203, 32, 64,201,232,224
267 DATA   5,208,245,162,  7,189,192,  2
268 DATA  32, 64,201,202, 16,247, 24, 96
269 DATA 224,213,144,  1,202,138, 41,127
270 DATA 201, 80,144,  2,233,  3,201, 69
271 DATA 144,  2,233,  3,201, 65,144,  2
272 DATA 233, 33,201, 28,144,  2,233,  8
273 DATA 201, 17,144,  2,233, 11, 56,233
274 DATA   4,170,169, 60, 32, 64,201,160
275 DATA 255,202,240,  8,200,185,122,203
276 DATA  16,250, 48,245,200,185,122,203
277 DATA  48,  6, 32, 64,201, 24,144,244
278 DATA  41,127, 32, 64,201,169, 62, 76
279 DATA  64,201, 87, 72,212, 67, 82,196
280 DATA  82, 79,206, 72, 79,205, 82, 69
281 DATA 196, 67, 82,210, 71, 82,206, 66
282 DATA  76,213, 79, 82,199, 70,177, 70
283 DATA 179, 70,181, 70,183, 70,178, 70
284 DATA 180, 70,182, 70,184, 66, 76,203
285 DATA  67, 82,213, 82, 79,198, 67, 76
286 DATA 210, 66, 82,206, 72, 82,212, 71
287 DATA  82,177, 71, 82,178, 72, 71,206
288 DATA  72, 66,204, 71, 82,179, 80, 85
289 DATA 210, 67, 82,204, 89, 69,204, 67
290 DATA  89,206, 27, 42,  4,  8,  0

READY.
```

Basic-Lader für das Treiberprogramm (Schluß)

```
Druck mit OPEN 17,17,0:

100 POKE 53280,6: POKE 53281,6
110 PRINT CHR$(14)"";
120 PRINT CHR$(14)"";
130 PRINT CHR$(14)"****************************"
140 PRINT "";
150 PRINT "*                          *"
160 PRINT "";
170 PRINT "*   —64 |REIBERPROGRAMM    *"
180 PRINT "";
190 PRINT "*                          *"
200 PRINT "";
210 PRINT "*   FUER DEN  NSCHLUSS     *"
220 PRINT "";
230 PRINT "*  VON        -RUCKER      *"
240 PRINT "";
250 PRINT "*       AM       - ORT     *"
260 PRINT "";
270 PRINT "*                          *"
280 PRINT "";
290 PRINT "****************************"
```

Bild 4. Musterlisting, erstellt auf einem Epson FX-80

Drucker verwendbar, sofern auf die Ausgabe von commodoreeigenen Grafikzeichen verzichtet wird. Für diese Funktion muß der Drucker jedoch vorübergehend in den Bitmustermodus umgeschaltet werden, was während einer normalen Textausgabe möglich sein muß. Diese Umschaltung erfolgt im Programm in der Schleife ab Adresse $CB10. Die Anzahl der auszugebenden Steuerzeichen ist unter Adresse $CB18 gespeichert und die Steuerzeichen selbst stehen ab Adresse $CBD2 am Ende des Programms. Für einen Epson-Drucker wird die Folge:

ESC »*« 4 8 0

ausgegeben, wobei mit vier der Bitmustermodus »CRT-Grafik« ausgewählt wird, 8 das niederwertige Byte und 0 das höherwertige Byte der Anzahl auszugebender Punktmatrixspalten darstellt. Der Epson-Drucker kehrt nach der Ausgabe der spezifizierten Anzahl Musterbytes wieder in den Textmodus zurück.

(H.Eyssele)

Mitmachen, aber wie?

Die 64'er-Redaktion freut sich über jeden Beitrag unserer Leser. Die Erfahrungen bei unseren Schwesterzeitschriften haben aber gezeigt, daß viele Einsender nicht genau wissen, in welcher Form sie ihre Manuskripte einsenden sollen. Die unten aufgeführten Punkte stellen keine »Richtlinien« dar. Dennoch sollte sich jeder, der ein Programm oder einen Artikel einsenden will, an ein gewisses Schema halten. Dies erleichtert zum einen die Arbeit der Redaktion, zum anderen kommt es auch Ihnen selbst zugute, da wir vollständige Listings oder Artikel schneller veröffentlichen können. Folgende Kriterien sind also generell zu beachten.

1. Auf der ersten Seite des Anschreibens sollten der Name, die vollständige Anschrift mit Telefonnummer sowie das Einsendedatum stehen.

2. In der »Betreffzeile« tragen Sie die genaue Spezifikation des verwendeten Computers und falls erforderlich, die Basic-, ROM- oder DOS-Versionen sowie die Speicherkonfigurationen ein. Der Titel des Artikels sollte ebenfalls daraus ersichtlich sein (auch für eventuelle Nachträge).

3. Im darauffolgenden Text können Sie Wesentliches zu Ihrer Person, zur Entstehungsgeschichte des Programms/Artikels, der Absicht, der Vorteile gegenüber anderen Programmen oder Methoden, der Eigenschaften und so weiter erläutern.

4. Auf der nächsten Seite beginnt die eigentliche Programmbeschreibung. Diese sollte nach Möglichkeit mit der Schreibmaschine geschrieben werden oder als Computerausdruck vorliegen. Den Text bitte mit mindestens eineinhalb oder doppeltem Zeilenabstand verfassen. Am linken und rechten Rand mindestens drei Zentimeter Freiraum für Korrekturen und Bemerkungen lassen.

5. Diese und alle nachfolgenden Seiten sollten durchnumeriert sein und in der Kopfzeile jeweils den Titel des Programms und den Namen des Autors enthalten.

6. Der Überschrift des Artikels schließen sich zwei oder drei einleitende Sätze an, welche die wesentlichen Punkte des Textes zusammenfassen.

Der Text selbst sollte in etwa folgenden Aufbau aufweisen:

— Angaben auf welchem Computer das Programm lauffähig ist sowie welche Erweiterungen und Peripherie notwendig sind

— ausführliche Beschreibung der Programmfunktion (mit Verweisen auf Ein-/Ausgabebeispielen wie Grafiken, Bildschirmfotografien, Hardcopys oder Diagrammen)

— detaillierte Programmbeschreibung (mit Verweisen auf Programmablaufplan, Variablendefinition, Startadressen der einzelnen Unterprogramme, Beschreibung wichtiger Programmzeilen etc.)

— eventuelle Umsetzung auf andere Basic-Dialekte oder Computer

7. Die genauen Lade- und Abspeicherschritte des Programms und der im Programm vorkommenden Routinen sollten dokumentiert sein.

8. Listings aus reprotechnischen Gründen nur als Original (keine Kopien) auf weißem, unliniertem Papier mit neuwertigem Farbband gedruckt einsenden. In den Listings dürfen grundsätzlich keine handschriftlichen Eintragungen stehen.

9. In den Kopfzeilen des Programms bitte den Titel desselben, die Computerkonfiguration, den eigenen Namen und die Adresse mit Telefonnummer eintragen (es soll vorkommen, daß sich Listings und Manuskripte verselbständigen, und mit beiden allein läßt sich wenig anfangen).

REM-Zeilen im Programm dienen der Übersichtlichkeit und sollten, falls nicht speicherkritische Aspekte dagegensprechen, immer zur Strukturierung eingesetzt werden (siehe u. a. »Sauberes Programmieren«).

10. Um das Eintippen für andere zu erleichtern, sollten Sie CHR$(X)-Werte und TAB(X) oder SPC(X) anstatt Cursor-Manipulationen für die Ausgabeformatierung verwenden. So ist die Befehlssequenz FOR I=1 TO 6:PRINT:NEXT zur Erzeugung von sechs Carriage Returns leichter einzutippen und auf andere Basic-Computer wesentlich einfacher zu übertragen. Und ist es nicht auch übersichtlicher statt einem Dutzend Cursor-Rechts-Symbolen einfach SPC(12) zu benutzen? Überprüfen Sie Ihr Programm einmal hinsichtlich dieser »Kleinigkeiten«.

11. Da wir (in Ihrem eigenen Interesse) nur getestete Programme veröffentlichen wollen, legen Sie bitte unbedingt eine Diskette oder Kassette, auf der das betreffende Programm mit mindestens einer Sicherheitskopie abgespeichert ist, bei. Auf der Diskette/Kassette und deren Umhüllung unbedingt den Namen mit vollständiger Adresse und Computerbezeichnung vermerken.

12. Wollen Sie mehrere Programme/Artikel gleichzeitig einsenden, so trennen Sie die Programme/Artikel nach dem oben aufgezeigten Schema. Die Einsendung mehrerer Disketten/Kassetten ist hingegen nicht notwendig.

13. Artikel können beliebig lang sein — von einzeiligen Routinen bis zu Serien über mehrere Ausgaben. Ein durchschnittlicher Artikel hat rund vier bis acht Schreibmaschinenseiten.

14. Hardcopys, Flußdiagramme, Zeichnungen und Bildschirmfotos dienen der Anschaulichkeit. Sie sollten nach Möglichkeit nicht fehlen. Zu jedem der vorgenannten »Zugaben« gehört aber eine Bildunterschrift und ein Verweis im Text.

15. Programme/Artikel die unserem Verlag zur Veröffentlichung angeboten werden, sollten aus urheberrechtlichen Gründen nicht gleichzeitig einem anderem Verlag vorliegen.

16. Das 64'er Magazin zahlt für Listings eine Pauschale zwischen 100 und 300 Mark. Für reine Artikel beträgt das Honorar zwischen 0,80 und 1,00 Mark pro Druckzeile. Für Disketten/Kassetten werden 30 Mark extra berechnet.

17. Sollten sich nach Erhalt eines positiven Anwortschreibens noch irgendwelche Änderungen oder Verbesserungen des Programms ergeben haben, teilen Sie uns das bitte umgehend mit. In diesem Falle benötigen wir ein vollständig neues Listing mit entsprechendem Datenträger.

(aa)

Wie schicke ich meine Programme ein?

kurvendiskussion in hires-grafik mit hardc

Mit dem Programm »Kurvendiskussion« besitzen Sie die Möglichkeit, eine komplette Funktionsanalyse durchzuführen. Sie können die Funktion auf dem Bildschirm und auf dem Drucker 1526 (neues ROM) plotten lassen. Zusätzlich wird die X- und Y-Achse, mit Einheitsstrichen versehen, geplottet.

Zusätzlich können Sie folgende wichtige grafische Daten abrufen: Nullstellen, relative Minima und Maxima, das absolute Minimum und Maximum, die Fläche unter dem Graphen, das Volumen des Rotationskörpers und Sie können einzelne Funktionswerte abfragen. Diese grafischen Daten werden auf einige Stellen hinter dem Komma genau berechnet. In seltenen Fällen können diese auch falsch berechnet werden. Dies ist teilweise auf Fehler im Betriebssystem und auf Fehler bei der Approximation (Näherung) zurückzuführen.

Eingabe der Funktionsdaten

a) Eingabe der Funktion

Dies geschieht in der normalen Commodore-Schreibweise, wie sie in dem Handbuch auf den Seiten 23 bis 29 beschrieben ist. Beispiel: normale Darstellung:

F(X) = COSX + COS2X + COS5X

Eingabe in den Computer:

COS(X) + COS(2*X) + COS (5*X)

Mit dieser Schreibweise können Sie jede Funktion eingeben, die im Commodore-Basic implementiert ist: (ABS(X), ATN(X), COS(X), EXP(X), INT(X), LOG(X), SIN(X), SQR(X) und TAN(X)).

b) Eingabe des Intervalls, in der die Funktion geplottet werden soll:

Zuerst wird die linke Grenze des Intervalls eingegeben, danach die rechte durch ein Komma getrennt. Die grafischen Daten werden später nur für dieses von Ihnen angegebene Intervall berechnet. Beispiel: normale Darstellung:

[—10,10]

Eingabe in den Computer:

—10,10

Natürlich dürfen die Funktionen nur in definierten Intervallen eingegeben werden. Die Wurzelfunktion ist zum Beispiel nur im positiven Bereich definiert. Sie darf also nicht im Intervall von —5 bis 7 eingegeben werden.

c) Eingabe der Einheiten

Hier wird eine positive Zahl sowohl bei der X-Achse als auch bei der Y-Achse angegeben. Es können nur bis zu 30 Einheitenstriche auf der X-Achse und Y-Achse geplottet werden. Falls Sie keine Einheitenstriche benötigen, drücken Sie einfach »RETURN«.

d) Ausdruck auf dem 1526-Drucker (mit neuen ROMs)

Sie haben die Möglichkeit, die Funktion in verschiedenen Maßstäben zu drucken (1:1, 1:2, 1:4). Je nach Eingabe der Vergrößerung wird die Kurve entsprechend groß geplottet. Anschließend werden noch wichtige Angaben zur Funktion gedruckt. Falls Sie später noch grafische Daten abrufen werden, werden diese automatisch auch noch gedruckt. Wenn Sie keinen Ausdruck auf dem Drucker benötigen oder wenn Sie keinen 1526-Drucker besitzen, drücken Sie einfach »RETURN«.

e) Höchster Y-Wert

Das Programm ist so konzipiert, daß die Funktion immer den gesamten Grafikbildschirm voll ausnutzt. Wenn jetzt in dem angegebenen Intervall eine Unendlichkeitsstelle auftritt, wird die Funktion im Bereich der X-Achse sehr gestaucht. Dies soll durch die Angabe des höchsten Y-Werts verhindert werden. Wenn Sie zum Beispiel die Funktion 1/X im Intervall —5,5 plotten lassen und Sie geben als höchsten Y-Wert 10 an, dann werden alle Werte, die größer als 10 sind, weggelassen. Jetzt liegt der Wert F(X) = 10 am oberen Ende des Grafikbildschirms und F(X) = —10 am unteren Ende. Dadurch wird die Funktion im Bereich der X-Achse nicht gestaucht.

Falls der angegebene Y-Wert größer als der höchste Funktionswert ist, hat dieser keinen Einfluß auf den Ausdruck. Wenn Sie nur »RETURN« drücken, beträgt der höchste Y-Wert automatisch 1000.

Abruf der grafischen Daten

Nachdem die Funktion korrekt eingegeben wurde, braucht der Computer zirka 2 bis 3 Minuten, bis er mit dem Plotten fertig ist. Danach drücken Sie bitte eine beliebige Taste. Es erscheinen einige wichtige Informationen und ein Menü. Auf Druck der entsprechenden Zahl wird das Erwünschte ausgeführt. Wir wollen uns jetzt nur der zweiten Möglichkeit zuwenden, da sich die anderen wohl selbst erklären. Auf Druck der Taste 2 erscheint ein weiteres Menü. Jetzt können Sie alle grafischen Daten errechnen lassen. Diese werden, wie unter Punkt 2.b erwähnt, nur in dem angegebenen Intervall, mit Ausnahme der speziellen Funktionswerte, errechnet. Folgende Sonderfälle sind zu beachten.

1. Absolute Extremwerte

Diese können nur korrekt berechnet werden, wenn der höchste Y-Wert größer ist als der größte Funktionswert.

2. Fläche unter dem Graphen

Der Computer gibt nicht wie bei der Integralrechnung die Differenz der Fläche zwischen dem Graphen und der X-Achse an, sondern den tatsächlichen absoluten Wert der Fläche.

3. Spezielle Funktionswerte

Falls Sie beim Abfragen eines Funktionswertes genau eine Unendlichkeitsstelle oder eine nicht definierte Stelle erwischt haben sollten, gibt der Computer eine Null als Funktionswert an.

(Jan Schaefer)

opy — Funktion

Programmbeschreibung

Programmaufschluesselung nach Zeilennummern

Zeilen	Beschreibung
0-9	Addresse des Autors
10-19	Funktionseingabe
25,26	Eingabe des Intervalls
27-29	Eingabe der Einheiten
30-56	Eingabe des Druckmasstabs
57-59	Eingabe des hoechsten Y-Werts
60-63	Abfrage, ob alle Eingaben korrekt sind
64-68	Poken der Maschinenspracheroutine
69-72	Errechnen der Position der Y-Achse
74	Variablenfelder werden definiert
75-86	alle Funktionwerte werden errechnet, um die Position der X-Achse zu bestimmen
87,88	der Graphikbildschirm wird geloescht und angeschaltet
89-95	die Geraden werden mit einer kurzen Maschinensprache-routine geplottet
96-98	Plotten der Pfeilspitzen
99-108	Plotten der Einheitenstriche
109-120	die Funktion wird auf dem Bildschirmgeplottet; waehrendessen werden die graphischen Daten bestimmt
112	Berrechnen der Flaeche und des Rotationskoerkers
114	Plotten eines einzelnen Punktes in Maschinensprache
117	Nullstellen werden festgestellt
118,119	relative Maxima bzw. Minima werden festgestellt
121-153	die Funktion wird auf dem Drucker ausgegeben
154-164	Drucken des ersten Menues
165-168	Abfrage der einzelnen Tasten
169	Funktion nochmal sehen
170	neue Funktion eingeben
171	Programmende
172-180	Drucken des zweiten Menues
181-183	Abfrage der einzelnen Tasten
184-213	Nullstellen werden berrechnet
189-197	Nullstellen werden mit dem Newton Verfahren genaehert
198-204	hier wird festgestellt, ob die angenommenen Nullstellen auch 'echt' sind
205-211	Nullstellen werden ausgedruckt
214-246	Berrechnung der relativen Extremwerte
214-230	Feststellen, ob Extremwerte korrekt sind
231-240	Extremwerte werden durch Intervallschachtelung genae-hert
241-246	Extremwerte werden gedruckt
247-284	absolute Extremwerte werden errechnet
247-269	Extremwerte werden durch Intervallschachtelung genae-hert
270-284	Feststellen ob Extremwerte korrekt sind und Ausdruck
285-293	Naeherung der Flaeche unter dem Graphen mittels der Tangententrapezregel
294-302	Naeherung des Volumen des Rotationskoerpers
303-317	Abfrage von speziellen Funktionswerten
318,319	allgemeine Plotroutine
320-360	Datas der Maschinenspracheroutinen

Listing »Kurvendiskussion«

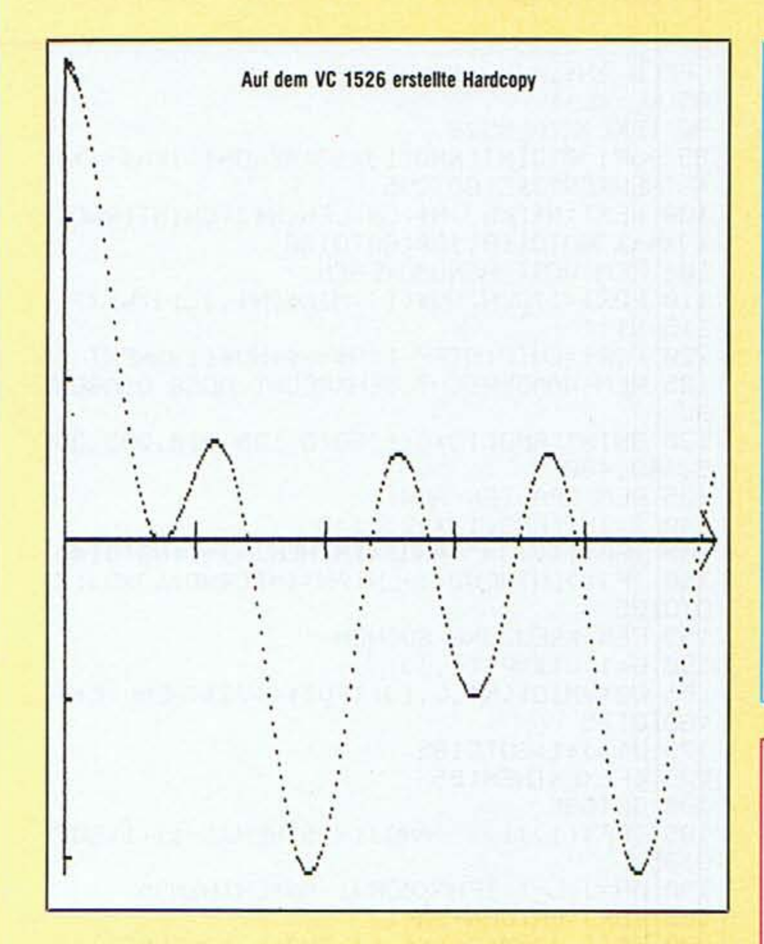
```
0 REM
1 REM     DIESES PROGRAMM WURDE 1983 ER-
2 REM     STELLT VON:
3 REM              *****************
4 REM              *JAN SCHAEFER    *
5 REM              *IN DER LOHN 9   *
6 REM              *                *
7 REM              *5100 AACHEN 1   *
8 REM              *TEL.02408/3640 *
9 REM              *****************
10 POKE53280,14:POKE53281,6:POKE646,1
11 PRINT"▒▒     K U R V E N D I S K U S
S I O N"
12 PRINT"▒              VON JAN SCHAEFER"
13 INPUT"▒FUNKTION F(X)=";A$
14 IFA$=""THENA$="1"
15 POKE646,6
16 PRINT"▒▒20DEFFNA(X)=";A$
17 PRINT"21A$="CHR$(34)A$CHR$(34)"
18 PRINT"RUN20▯▯▯▯";
19 POKE631,13:POKE632,13:POKE633,13:POKE
634,13:POKE198,4:END
20 DEFFNA(X)=X^2
21 A$="X^2"
22 POKE646,1:PRINT"▯▯▯▯▯":C$="
                                     "
23 FORI=1TO6:PRINTC$:NEXT
24 PRINT"▯▯▯▯▯▯▯▯"
25 INPUT"▒INTERVALL DER X-ACHSE (X1,X2)"
;I1,I2
26 IFI2<=I1THENPRINT"▯▯▯":GOTO25
27 INPUT"▒EINHEITEN DER X-ACHSE";EX
28 IFEX=0THENEX=I2-I1
29 INPUT"▒EINHEITEN DER Y-ACHSE";EY
30 PRINT"▒SOLL DIE FUNKTION GEDRUCKT WER
DEN (J/N)"
31 GETD$:IFD$=""THEN31
32 PRINT"▯▯▯":FORI=1TO3:PRINTC$:NEXT:PRI
NT"▯▯▯▯"
33 IFD$="J"THEN35
34 GOTO57
35 PD=1:E$="DIE FUNKTION WIRD"
36 PRINT"▒SOLL DIE FUNKTION IN X-RICHTUN
G VER-"
37 PRINT"▒GROESSERT WERDEN (J/N)?"
38 GETD$:IFD$=""THEN38
39 IFD$="N"THEN41
40 E$=E$+" IN X-RICHTUNG":VX=1
41 PRINT"▯▯▯▯▯":FORI=1TO5:PRINTC$:NEXT:P
RINT"▯▯▯▯▯▯"
42 PRINT"▒SOLL DIE FUNKTION IN Y-RICHTUN
G VER-"
43 PRINT"▒GROESSERT WERDEN (J/N)?"
44 GETD$:IFD$=""THEN44
45 IFD$="N"THEN49
46 IFVX=0THENE$=E$+" IN Y-RICHTUNG":GOTO
48
47 E$=E$+" UND":F$="IN Y-RICHTUNG "
48 VY=255:AR=51
49 IFVX=1ORVY=255THENF$=F$+"VERGROESSERT
 GEPLOTTET.":GOTO51
50 E$=E$+" GEPLOTTET."
51 PRINT"▯▯▯▯"
52 IFVY=0THENAR=25
53 FORI=0TO3:PRINTC$:NEXT
54 PRINT"▯▯▯▯"E$
55 IFF$=""THEN57
56 PRINT"▒"F$
57 INPUT"▒HOECHSTER Y-WERT";HX
58 IFHX=0THENHX=100
59 IFEY=0THENEY=HX
60 PRINT"▒SIND ALLE EINGABEN KORREKT (J/
N)"
61 GETD$:IFD$=""THEN61
62 IFD$="N"THEN11
63 PRINT"▯"C$
64 REM      POKEN DER MASCHINENROUTINE
65 IFMP=1THEN71
66 FORI=0TO64:READA:POKE828+I,A:NEXT
67 FORI=0TO651:READA:POKE49152+I,A:NEXT
68 POKE768,87:POKE769,168
69 REM      ERRECHNEN DER POSITION
70 REM             DER GERADEN
71 IB=I2-I1:IFI1>0THENYA=0:GOTO73
72 YA=INT(319/IB*ABS(I1)+.5)
73 N=IB/319
74 DIMK(325):DIMNU(321):DIMNR(100):DIMRE
(321):DIMRR(100):DIMRM(321)
75 WI=1E+38:WA=-WI
76 FORI=I1TOI2+NSTEPN
77 J=J+1:K(J)=FNA(I)
78 IFK(J)>WATHENWA=K(J):MA=I
79 IFK(J)<WITHENWI=K(J):MI=I
80 NEXT
81 IFWA>HXTHENWA=HX
82 IFWI<-HXTHENWI=-HX
83 IH=WA-WI
84 IFWI>0THENXA=199:IH=WA:WI=0:GOTO88
85 IFWA<0THENXA=0:IH=ABS(WI):GOTO88
86 XA=INT(199/IH*ABS(WA)+.5)
87 REM      STARTEN DES BITMAP-MODE
88 V=53248:AD=8192:SYS828:POKE16701,VY:P
OKEV+17,59:POKEV+24,24:POKE53280,1
89 REM      ZEICHNEN DER GERADEN
90 IFYA>319THENYA=319
91 IFYA<0THENYA=0
92 X=YA:Y=0:GOSUB318:POKE16704,PEEK(1672
2):SYS49575
93 IFXA>199THENXA=199
94 IFXA<0THENXA=0
95 Y=XA:X=0:GOSUB318:SYS49604
96 REM      ZEICHNEN DER PFEILSPITZEN
97 FORI=-7TO7:Y=XA+I:X=319-ABS(I):GOSUB3
18:NEXT
98 FORI=-7TO7:Y=ABS(I):X=YA+I:GOSUB318:N
EXT
99 REM      ZEICHNEN DER EINHEITEN
100 REM             DER X-ACHSE
101 D=(EX*319)/IB:IFD<10THEN106
102 FORI=YATO310STEPD:FORU=-4TO4:Y=XA+U:
X=I+.5:GOSUB318:NEXTU,I
103 FORI=YATO10STEP-D:FORU=-4TO4:Y=XA+U:
X=I+.5:GOSUB318:NEXTU,I
104 REM      ZEICHNEN DER EINHEITEN
105 REM             Y-ACHSE
106 D=(EY*199)/IH:IFD<6THEN110
107 FORI=XATO195STEPD:FORU=-3TO3:Y=I+.5:
X=YA+U:GOSUB318:NEXTU,I
108 FORI=XATO5STEP-D:FORU=-3TO3:Y=I+.5:X
=YA+U:GOSUB318:NEXTU,I
109 REM      ZEICHNEN DER FUNKTION
110 E=-199/IH:F=ABS(WI)*E+199
111 FORI=1TO319:X=I-1:Y=K(I)*E+F
112 FL=FL+ABS(K(I)):RO=RO+K(I)^2:RV=RV+K
(I)*K(I+1)
113 IFY>201ORY<0THEN115
114 SYS49629.Y+.5.X/8.XAND255:SYS49744.2
^PEEK(16722)
115 IFI<2THENNEXT
116 A=INT(Y+.5):D=K(I-1)*E+F:B=INT(D+.5)
117 IFB<XAANDA>=XAORB=>XAANDA=<XATHENM=M
+1:NU(M)=I*N+I1-N
```

```
118 C=K(I+1)*E+F:IFY>DANDY=>CTHENT=T+1:R
E(T)=I*N+I1-N:RM(T)=1:GOTO120
119 IFY<DANDY=<CTHENT=T+1:RE(T)=I*N+I1-N
:RM(T)=0
120 NEXT
121 REM     PLOTTEN DER FUNKTION
122 IFPD=0THEN154
123 OPEN5,4,5:OPEN4,4:OPEN6,4,6:SYS49152
124 PRINT#6,CHR$(20)
125 PRINT#4
126 PRINT#4,CHR$(14)CHR$(14)"     KURVEND
ISKUSSION"
127 PRINT#4
128 PRINT#4,CHR$(14)"            VON JAN
SCHAEFER"
129 PRINT#4
130 FORI=0TOAR:PRINT#6,CHR$(20):PRINT#4:
PRINT#6,CHR$(0)
131 FORU=0TO39
132 I$="":IFPEEK(16705)=255THEN141
133 FORJ=0TO7:A=PEEK(16709+J):I$=I$+CHR$
(A):NEXTJ
134 J$="":C$=""
135 PRINT#5,I$
136 FORK=0TOU:J$=J$+C$:C$=" ":NEXTK
137 IFVX=0THEN140
138 PRINT#4,CHR$(14)J$CHR$(254)
139 GOTO141
140 PRINT#4,J$CHR$(254)
141 B=0:SYS49160:NEXTU,I
142 PRINT:PRINT#6,CHR$(20)
143 PRINT#4,
144 PRINT#4,
145 PRINT#4,"F(X)="A$
146 PRINT#4,
147 PRINT#4,"INTERVALL VON "I1" BIS "I2
148 PRINT#4,
149 PRINT#4,"DIE EINHEITEN DER X-ACHSE S
IND IM ABSTAND VON"EX"GESETZT WORDEN."
150 PRINT#4,
151 PRINT#4,"DIE EINHEITEN DER Y-ACHSE S
IND IM ABSTAND VON"EY"GESETZT WORDEN."
152 PRINT#4
153 PRINT#4,"DER HOECHSTE Y-WERT IST"HX"
."
154 GETD$:IFD$=""THEN154
155 PRINT"[CLR]":POKEV+17,27:POKEV+24,21:POK
E53280,6:POKE53281,9
156 PRINT"F(X)="A$
157 PRINT"[DOWN]INTERVALL VON"I1"BIS"I2
158 PRINT"[DOWN]EINHEITEN DER X-ACHSE :"EX
159 PRINT"[DOWN]EINHEITEN DER Y-ACHSE :"EY
160 PRINT"[DOWN]DER HOECHSTE Y-WERT IST"HX"."
161 PRINT"[DOWN]1  DIE FUNKTION NOCHMAL SEHEN
"
162 PRINT"[DOWN]2  GRAPHISCHE DATEN DER FUNKT
ION"
163 PRINT"[DOWN]3  NEUE FUNKTION EINGEBEN"
164 PRINT"[DOWN]4  PROGRAMMENDE"
165 GETD$:IFD$=""THEN165
166 A=VAL(D$)
167 ONAGOTO169,172,170,171
168 GOTO165
169 POKE251,0:SYS862:POKEV+17,59:POKEV+2
4,24:POKE53280,1:GOTO154
170 CLR:POKE768,139:POKE769,227:MP=1:GOT
O10
171 POKE768,139:POKE769,227:END
172 POKE53281,2:POKE53280,3:PRINT"[CLR]";
173 PRINT"[DOWN]1  NULLSTELLEN
174 PRINT"[DOWN]2  RELATIVE EXTREMWERTE
175 PRINT"[DOWN]3  ABSOLUTE EXTREMWERTE
176 PRINT"[DOWN]4  FLAECHE UNTER DEM GRAPHEN
177 PRINT"[DOWN]5  VOLUMEN DES ROTATIONSKOERP
ERS
178 PRINT"[DOWN]6  SPEZIELLER FUNKTIONWERT
179 PRINT"[DOWN]7  RUECKSPRUNG INS HAUPTMENUE
180 PRINT"[DOWN]";
181 GETD$:IFD$=""THEN181
182 A=VAL(D$):POKE53280,1:POKE53281,14:O
NAGOTO184,215,247,286,295,303,155
183 POKE53281,2:POKE53280,3:GOTO181
184 REM     NULLSTELLENBERECHNUNG
185 R=0:PRINT"[CLR][DOWN]             NULLSTELLE
N"
186 IFPD=1THENPRINT#4:PRINT#4,"
  NULLSTELLEN"
187 IFM=0ANDPD=1THENPRINT#4:PRINT#4,"KEI
NE NULLSTELLEN"
188 IFM=0THENPRINT"[DOWN]KEINE NULLSTELLEN":G
OTO212
189 FORI=1TOM
190 N1=NU(I)
191 FORU=1TO10
192 S=(FNA(NU(I)+(10^-8))-FNA(NU(I)))/10
^-8
193 IFABS(S)<1E-38THENGOTO195
194 NU(I)=NU(I)-(FNA(NU(I)))/S
195 IFABS(N1-NU(I))>NORN1=NU(I)THENNU(I)
=-1000:GOTO197
196 NEXTU
197 NEXTI
198 FORJ=1TOI:FORU=J+1TOI
199 IFJ=ITHEN204
200 IFABS(NU(J)-NU(U))<NTHENNU(U)=-1000
201 NEXTU
202 IFNU(J)=-1000THEN204
203 R=R+1:NR(R)=NU(J)
204 NEXTJ
205 IFR=0ANDPD=1THENPRINT#4:PRINT#4,"KEI
NE NULLSTELLEN"
206 IFR=0THENPRINT"[DOWN]KEINE NULLSTELLEN":G
OTO212
207 FORI=1TOR
208 NR(I)=INT(NR(I)*100+.5)/100
209 IFPD=1THENPRINT#4:PRINT#4,I". FUER X
="NR(I)
210 PRINT"[DOWN]"I". FUER X="NR(I)
211 NEXT
212 GETD$:IFD$=""THEN212
213 GOTO172
214 REM     RELATIVE EXTREMWERTE
215 IFPD=1THENPRINT#4:PRINT#4,"
 RELATIVE EXTREMWERTE"
216 PRINT"[CLR][DOWN]          RELATIVE EXTREMWER
TE"
217 IFT=0ANDPD=1THENPRINT#4:PRINT#4,"KEI
NE RELATIVEN EXTREMWERTE"
218 IFT=0THENPRINT"[DOWN]KEINE RELATIVEN EXTR
EMWERTE":GOTO212
219 TS=T:FORI=1TOT
220 A=FNA(RE(I))
221 B=FNA(RE(I)-N)
222 C=FNA(RE(I)+N)
223 IFABS(A-B)>IH/4THENTS=TS-1:RE(I)=-10
00
224 IFABS(A-C)>IH/4THENTS=TS-1:RE(I)=-10
00
225 NEXT
226 IFTS=0THENT=0:GOTO217
227 FORI=1TOT
228 IFRE(I)=-1000THEN230
```

Listing »Kurvendiskussion« (Fortsetzung)

Listing »Kurvendiskussion« (Fortsetzung)

```
229 TT=TT+1:RE(TT)=RE(I)
230 NEXT:T=TT:IFTT=0THEN217
231 FORI=1TOT:FORU=0TO6
232 B=N/10^U
233 FORJ=1TO10
234 D=FNA(RE(I)+B)
235 E=FNA(RE(I))
236 IFD>EANDRM(I)=0THENRE(I)=RE(I)+B:NEX
TJ
237 IFD<EANDRM(I)=0THENRE(I)=RE(I)-B:NEX
TJ
238 IFD<EANDRM(I)=1THENRE(I)=RE(I)+B:NEX
TJ
239 IFD>EANDRM(I)=1THENRE(I)=RE(I)-B:NEX
TJ
240 NEXTU,I
241 FORI=1TOT
242 A=FNA(RE(I))
243 A=INT(A*1000+.5)/1000:RE(I)=INT(RE(I
)*1000+.5)/1000
244 PRINT"■"I". FUER F("RE(I)")="A
245 IFPD=1THENPRINT#4:PRINT#4,I". FUER F
("RE(I)")="A
246 NEXT:GOTO212
247 REM      ABSOLUTE EXTREMWERTE
248 PRINT"■■            ABSOLUTE EXTREMWER
TE"
249 IFPD=1THENPRINT#4:PRINT#4,"
 ABSOLUTE EXTREMWERTE"
250 AE(1)=MA:AE(2)=MI
251 FORI=1TO2:FORU=1TO6
252 B=N/10^U
253 FORJ=1TO10
254 D=FNA(AE(I)+B)
255 E=FNA(AE(I))
256 IFD<EANDI=1THENAE(I)=AE(I)-B:NEXTJ
257 IFD>EANDI=1THENAE(I)=AE(I)+B:NEXTJ
258 IFD<EANDI=2THENAE(I)=AE(I)+B:NEXTJ
259 IFD>EANDI=2THENAE(I)=AE(I)-B:NEXTJ
260 NEXTU,I
261 IFAE(1)>I2THENAE(1)=I2
262 IFAE(1)<I1THENAE(1)=I1
263 IFAE(2)<I1THENAE(2)=I1
264 IFAE(2)>I2THENAE(2)=I2
265 FORI=1TO2
267 A(I)=FNA(AE(I))
268 A(I)=INT(A(I)*1000+.5)/1000:AE(I)=IN
T(AE(I)*1000+.5)/1000
269 NEXT
270 IFABS(A(1))>HXTHENPRINT"■DAS ABSOLUT
E MAXIMUM KANN NICHT BESTIMMT■WERDEN."
271 IFABS(A(1))>HXANDPD=1THENPRINT#4,
272 IFABS(A(1))>HXANDPD=1THENPRINT#4,"DA
S ABSOLUTE MAXIMUM KANN NICHT BESTIMMT W
ERDEN."
273 IFABS(A(1))>HXTHEN277
274 PRINT"■DAS ABSOLUTE MAXIMUM LIEGT BE
I"
275 PRINT"■F("AE(1)")="A(1)
276 IFPD=1THENPRINT#4:PRINT#4,"DAS ABSOL
UTE MAXIMUM LIEGT BEI F("AE(1)")="A(1)
277 IFABS(A(2))>HXTHENPRINT"■DAS ABSOLUT
E MINIMUM KANN NICHT BESTIMMT■WERDEN."
278 IFABS(A(2))>HXANDPD=1THENPRINT#4
279 IFABS(A(2))>HXANDPD=1THENPRINT#4,"DA
S ABSOLUTE MINIMUM KANN NICHT BESTIMMT W
ERDEN."
280 IFABS(A(2))>HXTHEN212
281 PRINT"■DAS ABSOLUTE MINIMUM LIEGT BE
I"
282 PRINT"■F("AE(2)")="A(2)
283 IFPD=1THENPRINT#4:PRINT#4,"DAS ABSOL
UTE MINIMUM LIEGT BEI F("AE(2)")="A(2)
284 GOTO212
285 REM      FLAECHENBERECHNUNG
286 PRINT"■■           FLAECHE UNTER DEM GRA
PHEN"
287 IFPD=1THENPRINT#4:PRINT#4,"          FL
AECHE UNTER DEM GRAPHEN"
288 FL=FL-K(1)
289 F1=(K(1)+K(321))/2
290 F2=(F1+FL)*N:F2=INT(100*F2+.5)/100
291 PRINT"■ A="F2
292 IFPD=1THENPRINT#4:PRINT#4," A="F2
293 GOTO212
294 REM      ROTATIONSKOEPERBERECHNUNG
295 PRINT"■■        VOLUMEN DES ROTATIONSKO
ERPERS"
296 IFPD=1THENPRINT#4:PRINT#4,"        VOLU
MEN DES ROTATIONSKOERPERS"
297 RO=(RO-K(1)^2)/2+(K(321)^2+K(1)^2)/4
+(RV)/2
298 RO=RO*π*N
299 RO=INT(RO*100+.5)/100
300 PRINT"■ V="RO
301 IFPD=1THENPRINT#4:PRINT#4," V="RO
302 GOTO212
303 REM      SPEZIELLE FUNKTIONSWERTE
304 PRINT"■■           SPEZIELLE FUNKTIONSW
ERTE"
305 IFPD=1THENPRINT#4:PRINT#4,"           S
PEZIELLE FUNKTIONWERTE"
306 PRINT"■FUER WELCHEN PUNKT SOLL DER F
UNKIONS-"
307 INPUT"■WERT ERRECHNET WERDEN";A
308 B=FNA(A):B=INT(1000*B+.5)/1000
309 PRINT"■ F("A")="B
310 IFPD=1THENPRINT#4:PRINT#4," F("A")="
B
311 PRINT"■NOCH EINEN ?"
312 GETD$:IFD$=""THEN312
313 IFD$="N"THEN172
314 PRINT"□□□":FORI=0TO2:PRINTC$:NEXT:PR
INT"□□□":GOTO306
315 PRINT"■IN DEM PUNKT"A"KANN KEIN FUNK
TIONSWERT BESTIMMT WERDEN !"
316 IFPD=1THENPRINT#4:PRINT"IN DEM PUNKT
"A"KANN KEIN FUNKTIONSWERT BESTIMMT WERD
EN !"
317 GOTO311
318 IFY>201ORY<0ORX<0ORX>319THENRETURN
319 SYS49629,Y,X/8,XAND255:SYS49744,2^PE
EK(16722):RETURN
320 DATA169,32,133,252,169,0,133,251,169
,0,162,0,160,255,145,251,136,145,251
321 DATA240,3,76,76,3,230,252,232,224,34
,240,3,76,72,3,169,4,133,252,169,1,162
322 DATA0,160,255,145,251,136,145,251,24
0,3,76,106,3,230,252,232,224,4,240,3
323 DATA76,102,3,96
324 DATA169,32,133,252,169,0,133,251,160
,8,169,0,153,68,65,136,208,250,160
325 DATA0,162,8,177,251,141,60,65,173,60
,65,74,141,60,65,144,9,189,68,65,24
326 DATA105,128,157,68,65,202,208,235,17
4,61,65,224,255,240,1,200,162,8,177
327 DATA251,141,60,65,173,60,65,74,141,6
0,65,144,9,189,68,65,24,105,64,157
328 DATA68,65,202,208,235,200,162,8,177,
251,141,60,65,173,60,65,74,141,60,65
329 DATA144,9,189,68,65,24,105,32,157,68
,65,202,208,235,174,61,65,224,255,240
```


Auf dem VC 1526 erstellte Hardcopy

```
330 DATA1,200,162,8,177,251,141,60,65,17
3,60,65,74,141,60,65,144,9,189,68,65
331 DATA24,105,16,157,68,65,202,208,235,
200,162,8,177,251,141,60,65,173,60
332 DATA65,74,141,60,65,144,9,189,68,65,
24,105,8,157,68,65,202,208,235,174
333 DATA61,65,224,255,240,1,200,162,8,17
7,251,141,60,65,173,60,65,74,141,60
334 DATA65,144,9,189,68,65,24,105,4,157,
68,65,202,208,235,200,162,8,177,251
335 DATA141,60,65,173,60,65,74,141,60,65
,144,9,189,68,65,24,105,2,157,68,65
336 DATA202,208,235,174,61,65,224,255,24
0,1,200,162,8,177,251,141,60,65,173
337 DATA60,65,74,141,60,65,144,9,189,68,
65,24,105,1,157,68,65,202,208,235,24
338 DATA165,251,105,8,133,251,165,252,10
5,0,133,252,174,61,65,224,255,240,3
339 DATA76,141,193,174,62,65,224,255,240
,11,169,0,133,253,133,254,169,255,141
340 DATA62,65,24,165,253,105,8,133,253,1
65,254,105,0,133,254,166,253,224,64
341 DATA208,58,166,254,224,1,208,52,56,1
65,251,233,60,133,251,165,252,233,1
342 DATA133,252,169,0,133,253,133,254,17
4,63,65,224,255,208,21,169,0,141,63
343 DATA65,24,165,251,105,56,133,251,165
,252,105,1,133,252,76,141,193,169,255
344 DATA141,63,65
345 DATA160,8,136,185,69,65,201,0,208,10
,192,0,208,244,169,255,141,65,65,96
346 DATA169,0,141,65,65,96
347 DATA162,25,173,64,65,160,8,136,145,2
51,208,251,24,165,251,105,64,133,251
348 DATA165,252,105,1,133,252,202,208,23
0,96
349 DATA162,40,160,0,169,255,145,251,24,
```

Listing »Kurvendiskussion« (Ende)

```
165,251,105,8,133,251,165,252,105,0
350 DATA133,252,202,208,236,96
351 DATA32,253,174,32,158,183,142,76,65,
32,253,174,32,158,183,142,77,65,32
352 DATA253,174,32,158,183,142,82,65,173
,76,65,41,248,141,78,65,169,40,141
353 DATA79,65,32,99,194,173,76,65,41,7,2
4,109,80,65,133,251,173,81,65,105,0
354 DATA133,252,173,77,65,141,78,65,169,
8,141,79,65,32,99,194,24
355 DATA173,80,65,101,251,133,251,173,81
,65,101,252
356 DATA133,252,24,165,252,105,32,133
357 DATA252,173,82,65,41,7,141,82,65,169
,7,56,237,82,65,141,82,65,96,32,253
358 DATA174,32,158,183,142,82,65,160,0,1
77,251,13,82,65,145,251,96,169,0,141
359 DATA80,65,141,81,65,162,9,24,202,110
,81,65,110,80,65,224,0,240,18,110,78
360 DATA65,144,240,173,81,65,24,109,79,6
5,141,81,65,76,110,194,96
READY.
```

Ladeprogramm »Kurvendiskussion«

```
10 POKE16900,0:POKE16899,0:POKE16898,0:P
OKE43,4:POKE44,66
20 PRINT"▒NEW"
30 PRINT"▒▒LOAD"CHR$(34)"KURVENDISKUSSIO
N"CHR$(34)",8"
40 PRINT"▒▒▒▒▒RUN▒"
50 POKE631,13:POKE632,13:POKE633,13:POKE
198,3
READY.
```

```
Liste der wichtigsten Variablen
-------------------------------
I1:          linke Intervallgrenze
I2:          rechte Intervallgrenze
EX:          Einheiten der X-Achse
EY:          Einheiten der Y-Achse
FNA(X):      Funktionsterm
PD:          Ausdruck auf Drucker Ja/Nein
VX:          Ausdruck Vergroessert in X-Richtung?
VY:          Ausdruck Vergroeßert in Y-Richtung?
HX:          Hoechster Y-Wert
YA:          Lage der Y-Achse
XA:          Lage der X-Achse
K(0-321):    Funktionswerte
NU(0-X):     Nullstellen
RE(0-X):     Relative Extremwerte
RM(0-X):     Relatives Maximum oder Minmum?
AE(1-2):     Absolute Extremwerte
MA:          Maximum
MI:          Minimum
V:           Basicadresse des VIC
AD:          Anfang des Graphikbildschirms
F2:          Flaeche unter dem Graphen
RO:          Volumen des Rotationskoerpers
A,B,C,D,E,F:    Feld, Wald und Wiesen-Variablen
I,U,K,J:     Schleifenvariablen
A$:          Funktionsterm
```

Rätsel — ein Knobelprogramm

Sicher kennen Sie die Rätsel, bei denen man aus einem Buchstabensalat bestimmte Wörter heraussuchen muß. Dies ist nicht immer einfach, zumal die Wörter auch rückwärts oder diagonal angeordnet sein können. Solche Rätsel können Sie nun mit dem VC 20 selbst erstellen.

Das Programm Rätsel bringt 440 Buchstaben bunt durcheinander auf den Bildschirm. In diesem Buchstabensalat sind Wörter versteckt, deren Anzahl zuvor bestimmt werden kann (Zeile 35).

Die Suche ist nicht einfach, weil die Wörter zufällig verteilt (Zeile 130) kreuz und quer, diagonal und zum Teil auch rückwärts geschrieben werden (Zeilen 100 bis 120 drehen dafür etwa jedes vierte Wort herum).

Richtig gefundene Wörter werden im Buchstabensalat schwarz gedruckt. Wer vor der Aufgabe kapituliert, kann sich mit der Taste x die Lösung zeigen lassen.

Das Programm schreibt die zu versteckenden Wörter zuerst in das Feld F$ (22,20). Zugleich wird im Feld FF$ (22,20) registriert, welche Zeichen belegt sind.

Der Programmlauf wird im übrigen aus den eingefügten REM-Zeilen verständlich.

(Jürgen Curdt)

```
1 REM"s"=CLEAR HOME        "S"=CORSOR HOME
2 REM"q"CURSOR HOCH        "Q"=CURSOR RUNTER
3 REM"]"=CURSOR RECHTS "△"=CURSOR LINKS
4 REM "p"=SCHWARZ "E"=WEISS "£"=ROT
10 REM WOERTERSUCHE IM BUCHSTABENSALAT
15 REM JUERGEN CURDT,KESSEMEIERWEG 5,493
 DETMOLD,TEL 67264
20 POKE36879,93:PRINT"sE";:FORI=0TO439:P
RINTCHR$(RND(1)*26+65);:NEXT
25 PRINT"SpIQCQH]]]VERSTECKE]QFQ△UQ△EQ△R
Q△":PRINT"]qqqqDICHQQKREUZ]]UNDqq△△△△△△△
△QQ△UQQ△R"
30 PRINT"WOERTER]AQUQCQH":PRINT"QQQQ]]]]
]DqIqAqGqOqNqAqLSQQQQQQQQQQQQQQQQQQ"
35 PRINT" ZAHL DER VERSTECKTEN  WOERTER
(2-20)";:INPUTK$
40 POKE1000,VAL(RIGHT$(K$,2))
45 K=PEEK(1000):DIMF$(22,20):DIMFF%(22,2
0):DIMN$(K+1):DIMNU$(20)
50 FORI=1TOK+1:N$(I)="0":NEXT
55 LE$="SQQQQQQQQQQQQQQQQQQQQ
                              qqq"
60 PRINTLE$"QQ BITTE ETWAS WARTEN !S";
65 REM EINSPRUNG IN DIE WOERTERLISTE
70 FORI=0TOINT(RND(1)*50):READN$:IFN$="*
**"THENRESTORE:GOTO70
75 NEXT
80 REM WOERTER WAEHLEN - N$ - UND IN DAS
 FELD -N$,X,Y- SCHREIBEN
85 KL=KL+1
90 IFKL>KTHEN520
95 FORI=0TOINT(RND(1)*5):READN$:IFN$="**
*"THENRESTORE:GOTO95
100 NEXT:N$(KL)=N$:LN=LEN(N$):ONINT(RND(
1)*4+1)GOTO110,130:GOTO130
105 REM WORT HERUMDREHEN
110 FORI=1TOLN:NU$(I)=MID$(N$,I,1):NEXT
115 N$=""
120 FORI=LNTO1STEP-1:N$=N$+NU$(I):NEXT
125 REM WAAGERECHT,SENKRECHT ODER DIAGON
AL
130 ONINT(RND(1)*6+1)GOTO 135,210,285,36
5,440,480
135 REM SPALTEN ABW.
140 I=INT(RND(1)*22):J=0
145 IFF$(I,J)=""ANDJ<19THENJ=J+1:GOTO145
150 IFJ=>LNTHEND=J-LN:AN=INT(RND(1)*D):G
OTO195
155 REM KREUZUNG SUCHEN
160 L=1:V1$=F$(I,J)
165 V2$=MID$(N$,L,1):IFV1$<>V2$THENL=L+1
:GOTO175
170 J1=J+1:GOTO185
175 IFL<LNTHEN165
180 GOTO95
185 IFF$(I,J1)=""ANDJ1<19THENJ1=J1+1:GOT
O185
190 AN=J-L+1:IFAN<0ORJ1-AN<LNTHEN95
195 FORJ=ANTOAN+LN-1
200 F$(I,J)=MID$(N$,(J-AN)+1,1):FF%(I,J)
=KL:NEXT
205 GOTO80
210 REM SPALTEN AUFW.
215 I=INT(RND(1)*22):J=19
220 IFF$(I,J)=""ANDJ>0THENJ=J-1:GOTO220
225 IF19-J>=LNTHEND=20-J-LN:AN=INT(RND(1
)*D)+J+1:GOTO270
230 REM KREUZUNG SUCHEN
235 L=1:V1$=F$(I,J)
240 V2$=MID$(N$,L,1):IFV1$<>V2$THENL=L+1
:GOTO250
245 J1=J-1:GOTO260
250 IFL<LNTHEN240
255 GOTO95
260 IFF$(I,J1)=""ANDJ1>0THENJ1=J1-1:GOTO
260
265 AN=J-L+1:IFAN<0ORAN+LN>19ORJ-J1<LTHE
N95
270 FORJ=ANTOAN+LN-1
275 F$(I,J)=MID$(N$,(J-AN)+1,1):FF%(I,J)
=KL:NEXT
280 GOTO80
285 REM ZEILEN N.R.
290 J=INT(RND(1)*19):I=0
295 IFF$(I,J)=""ANDI<21THENI=I+1:GOTO295
300 IFI=>LNTHEND=I-LN:AN=INT(RND(1)*D):G
OTO350
305 REM KREUZUNG SUCHEN
310 L=1:V1$=F$(I,J)
315 V2$=MID$(N$,L,1):IFV1$<>V2$THENL=L+1
:GOTO325
320 GOTO335
325 IFL<LNTHEN315
330 GOTO95
335 I1=I+1
340 IFF$(I1,J)=""ANDI1<21THENI1=I1+1:GOT
O340
```

```
345 AN=I-L+1:IFAN<0ORI1-AN<LNTHEN95
350 FORI=ANTOAN+LN-1
355 F$(I,J)=MID$(N$,(I-AN)+1,1):FF%(I,J)
=KL:NEXT
360 GOTO80
365 REM ZEILEN N.L.
370 J=INT(RND(1)*20):I=21
375 IFF$(I,J)=""ANDI>0THENI=I-1:GOTO375
380 IF21-I>=LNTHEND=22-I-LN:AN=INT(RND(1
)*D)+I+1:GOTO425
385 REM KREUZUNG SUCHEN
390 L=1:V1$=F$(I,J)
395 V2$=MID$(N$,L,1):IFV1$<>V2$THENL=L+1
:GOTO405
400 J1=I-1:GOTO415
405 IFL<LNTHEN395
410 GOTO95
415 IFF$(J1,J)=""ANDJ1>0THENJ1=J1-1:GOTO
415
420 AN=I-L+1:IFAN<0ORAN+LN>21ORI-J1<LTHE
N95
425 FORI=ANTOAN+LN-1
430 F$(I,J)=MID$(N$,(I-AN)+1,1):FF%(I,J)
=KL:NEXT
435 GOTO80
440 REM DIAG.LU-RO
445 J=INT(RND(1)*(19-LN))+LN:AN=INT(RND(
1)*(23-LN)):J2=J:I2=AN
450 IFJ2>0ANDI2<22ANDF$(I2,J2)=""THENJ2=
J2-1:I2=I2+1:GOTO450
455 IFJ-J2<LNTHEN135
460 FORI=ANTOAN+LN-1
465 F$(I,J)=MID$(N$,I-AN+1,1):FF%(I,J)=K
L:J=J-1
470 NEXT
475 GOTO80
480 REM DIAG.LO-RU
485 J=INT(RND(1)*(20-LN)):AN=INT(RND(1)*
(23-LN)):J2=J:I2=AN
490 IFJ2<20ANDI2<22ANDF$(I2,J2)=""THENJ2
=J2+1:I2=I2+1:GOTO490
495 IFJ2-J<LNTHEN285
500 FORI=ANTOAN+LN-1
505 F$(I,J)=MID$(N$,I-AN+1,1):FF%(I,J)=K
L:J=J+1
510 NEXT
515 GOTO80
520 REM RAETSEL AUSGEBEN
525 PRINT"SE"::FORJ=0TO19:FORI=0TO21
530 IFF$(I,J)=""THENF$(I,J)=CHR$(RND(1)*
26+65)
535 PRINTF$(I,J);
540 NEXTI,J
545 REM EINGABE ABWARTEN
550 PRINTLE$"Q pSUCHE"K"WOERTER        UND
 TIPPE SIE EIN!S"::POKE198,0
555 EI$="":WAIT198,1:PRINTLE$
560 GETE$:IFE$=""THEN560
565 IFE$=CHR$(13)THEN585
570 IFE$="*"THEN665
575 IFE$=CHR$(20)THENEI$=LEFT$(EI$,LEN(E
I$)-1):PRINT"▲ ▲"::GOTO560
580 EI$=EI$+E$:PRINTE$;:GOTO560
585 FORKL=1TOK:IFEI$<>N$(KL)THENNEXT:GOT
O595
590 GOTO605
595 PRINTLE$:PRINT" LEIDER NICHT RICHTIG
  LOESUNG..............*";
600 GOTO555
605 R=R+1
610 REM GEFUNDENES WORT SCHWARZ SCHREIBE
N
615 PRINT"S"::FORJ=0TO19:FORI=0TO21
620 IFFF%(I,J)=KLTHENPRINTF$(I,J);:GOTO6
30
625 PRINT"1";
630 NEXTI,J:IFR=KTHEN645
635 PRINTLE$:PRINTR+1;"▲. WORT?S";
640 N$(KL)="":GOTO555
645 PRINTLE$" *** FEIN GEMACHT ***
NEUES SPIEL ?";
650 POKE198,0:WAIT198,1
660 PRINTLE$;:RUN45
665 REM LOESUNG GEBEN
670 PRINT"Sp"::FORJ=0TO19:FORI=0TO21
675 IFFF%(I,J)>0THENPRINTF$(I,J);:GOTO685
680 PRINT"1";
685 NEXTI,J
690 PRINT" NEUES SPIEL ?          TASTE D
RUECKEN !";
695 GOTO650
700 DATAREGENWURM,UEBERFLUSS,VORGARTEN,A
PFELSAFT,FEUERWEHR,MORGENSTERN,ABENDROT
705 DATAAFFENHAUS,POLTERGEIST,LOGISCH,DO
RFPLATZ,GASTHAUS,STEINKOHLE,REGENSCHIRM

READY.

1000 DATA"***"
1010 REM ZWISCHEN 700 UND 999 KOENNEN BE
LIEBIGE WOERTER EINGEFUEGT WERDEN

READY.
```

Listing des Programms »Rätsel«

```
O Z F L K V L W G B L O K L Y J D R A T
H A B V B P O F N C R X N E V T G G Z H
H K G G I T D D U X L T I K O V L S N J
A C H A T P V B J I N I C R J H S E N B
L Q C X U G N T B S F N K E N E T S E W
K B J M V P N E T Y H V E F Q T G V U K
B B D I D A L M M T M V R Q S Z I B V Z
R L Q W M L E E G S B C X N T K J J G T
E P B A E S K N R I C S T U T B Y R E S
B L I J S F A H X V Z W E R C H F E L L
A D J E W W N O H C S I F N E T N I T U
R Q R P M G I B O X O D R E L H C S I T
T T J F F F N U P L E U C H T E R M S Q
E P Q N K S C A Y T J V K J H Z S G X Y
Y T F Z K Y H S D Y E L R N P U M Z F W
G E M D N H E R B S T L A U B V V J J E
A R F G E L N C M W I P O T D F F Y S A
P O L Z R C I W L G N S B M J Q K X C H
T G M P U C E S C H R E I B H E F T L V
P B Y K R Y A J X K O Y V E S G C G P T
DIAMANT KNICKER ZWERCHFELL
WESTEN FERKEL SAUBOHNE
TINTENFISCH TISCHLER ACHAT
SCHREIBHEFT KANINCHEN HERBSTLAUB
LIBELLE TRABER MESSER
LEUCHTER
WOERTERSUCHE IM BUCHSTABENSALAT
```

Hardcopy des Bildschirms als Beispiel. Die unten stehenden Wörter sind im Salat versteckt, versuchen Sie diese zu finden.



CROUS

Dieses Spiel zeigt die Möglichkeiten, die bereits in der Grundversion des VC 20 stecken. Obwohl von den Farbgrafik- und Soundmöglichkeiten ausgiebig Gebrauch gemacht wird, benötigt das Programm nur knapp 3 KByte RAM.

Vor dem Start einer neuen Spielrunde

Ein Action-Spiel für den VC 20

Feindliche Weltraumschiffe greifen die Erde an! Sie haben in diesem Spiel die Aufgabe, die Invasion abzuwehren. Dazu besitzen Sie ein Raumschiff, den »Croussaider«, das über der Erde schwebt. Unter ihm liegt eine moderne Stadt mit Hochhäusern, Fabriken und einem Atomkraftwerk. Am oberen Rand des Bildschirms erscheinen die Angreifer, deren Ziel es ist, in der Stadt zu landen und sie zu zerstören. Sie bewegen sich rasch auf die Stadt zu, so daß schnelle Reaktionen erforderlich sind, um eine Landung zu verhindern.

Hat man es geschafft, den Angreifern eine bestimmte Zeit zu widerstehen, erscheint eine neue, gefährlichere Welle von Invasoren mit anderem Aussehen.

Wenn man fünf verschiedene Angriffswellen übersteht, erscheint wieder die erste, mit dem Unterschied, daß die Angreiferzahl steigt. So ist immer für Abwechslung und Spannung gesorgt.

Der fortgeschrittene Spieler, dem das Spiel zu einfach ist, kann den Wert der Variablen Y in Zeile 10 vermindern.

Um das Spiel zu beschleunigen, wurden einige Routinen in Maschinensprache geschrieben. Die erreichte Punktzahl wird ständig angezeigt, und eine »High-Score«-Funktion hält das jeweils beste Ergebnis fest.

Doch damit genug der Vorrede. Tippen Sie doch jetzt einfach das Listing ab und versuchen Sie, die Invasion zu verhindern ...

(Eberhard Aust)

Zeilen	
10	Einlesen des Maschinenprogramms
12	Einlesen der Grafikzeichen
19 - 99	Spielvorbereitung
100 - 130	Hauptspielroutine
200 - 206	Score-Routine
250 - 292	Anzeige des Highscore
300 - 314	Schwierigkeitsgrad ändern
1000 - 1012	Maschinensprache-DATA
1100 - 1107	Grafik-DATA

Programmaufbau von »Croussaider«

Variable	
A	Raumschiffposition
Y	Angreiferwahrscheinlichkeit
J,K,L	Aussehen des Angreifers
SC	Score
HI	High-Score
E	Inhalt des Registers 37151
F	Inhalt des Registers 37152
N	Konstante 36876
D	Zufallszahl zwischen 1 und 2

Die wichtigsten Variablen

```
1 REM CROUSSAIDER
2 REM ===========
3 REM
4 REM
10 FORA=0TO164:READB:POKE828+A,B:NEXT:FORJ=0TO41:READK:POKE7440+J,K:NEXT
12 FORJ=0TO159:READK:POKE7168+J,K:NEXT:FORJ=0TO7:POKE7168+32*8+J,0:NEXT
19 POKE36879,9:PRINT"■■■■■■":A=8086:B=8078:C=8095:SC=0:X=0
20 POKE36869,255:POKE36878,15:POKE7432,8:POKE7433,0:Y=10:N=36876:LI=10:IFQQ=0THE
N280
75 FORRR=0TO44:PRINT"■                                  ";:NEXT:PRINT"■■
 "
80 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■LL P L   ■Q P Q■ LL PLL LLLPLLL■MM■ROLOR■LLLLLLL
";
90 PRINT"■NNNNNNNNNNNNNNNNNNNNNN■":POKE7680+505,14:POKE38400+505,5
91 POKEA-1,9:POKEA,10:POKEA+1,11:GOSUB300
100 E=PEEK(37151):POKE37154,127:F=PEEK(37152):POKE37154,255:D=INT(RND(1)*2)+1:PO
KEN,S+D
103 IF(EAND16)=0ANDA>BTHENPOKEA+1,32:POKEA,32:A=A-D:POKEA-1,9:POKEA,10:POKEA+1,1
1:POKEN,R
104 IF(FAND128)=0ANDA<CTHENPOKEA-1,32:POKEA,32:A=A+D:POKEA+1,11:POKEA,10:POKEA-1
,9:POKEN,T
```

Listing »Croussaider«

SAIDER

```
106 POKE36877,0:SYS903:IF(EAND32)=0THENPOKEA-22,8:POKE36877,252+D
111 IFINT(RND(1)*Y)=0THENI=INT(RND(1)*14)+7728:POKEI-1,J:POKEI,K:POKEI+1,L
120 SYS828:SYS828+51:IFPEEK(7432)<>8THENGOSUB200
121 IFPEEK(7433)<>0THENGOTO250
127 U=U+1:IFU=1THENSYS7440:IFTI>2000THENGOSUB300
129 PRINT"SER"SC:IFU=2THENSYS(7461):U=0
130 SYS903:GOTO100
200 FORH=0TO15:POKE36879,INT(RND(1)*256):NEXT:POKE36879,9
201 H=PEEK(7432):POKE36877,INT(RND(1)*90)+128:POKE7432,8:IFH=0THENSC=SC+10:RETUR
N
202 IFH=2ORH=4THENSC=SC+40:RETURN
203 IFH=3THENSC=SC+INT(RND(1)*120):RETURN
204 IFH=5THENSC=SC+75:RETURN
205 IFH=1THENSC=SC+20:RETURN
206 RETURN
250 PRINT"QQQQQQQQQQQQQQQQQQ              S  SSLSSSSMSQQSQQSSSSSLSS";
252 POKEN,0:POKEN+1,128:FORU=15TO0STEP-1:POKEN+2,U:FORUU=0TO40:POKEN+3,INT(RND(1
)*128):NEXT:NEXT
279 IFSC>HITHENHI=SC
280 POKE36879,9:PRINT"S           ER"HI:PRINT"SQQQQQ    CROUSSAIDERQQ          E.
AUST"
281 GETA$:QQ=1:IFA$=""THEN281
282 GOTO19
300 TI$="000000":X=X+1:S=220+X*3:R=S-10:T=S+10:IFX=6THENY=Y-1:X=1:IFY<3THENY=3
310 IFX=1THENJ=32:K=1:L=32:Z=Y:RETURN
311 IFX=2THENJ=0:K=32:L=0:Z=Y:RETURN
312 IFX=3THENJ=2:K=3:L=4:Z=Y*2:RETURN
313 IFX=4THENJ=1:K=0:L=1:Z=Y*1.5:RETURN
314 IFX=5THENJ=2:K=3:L=4:Z=Y*1.5:RETURN
1000 DATA162,228,189,0,31,201,8,16,8,157,22,31,169,32,157,0,31,224,0,240,4,202,7
6,62,3
1001 DATA162,233,189,22,30,201,8,16,8,157,44,30,169,32,157,22,30,224,0,240,4,202
,76,87,3
1002 DATA96,162,0,189,184,31,201,8,48,9,224,43,240,4,232,76,113,3,96,142,9,29,76
,120,3
1003 DATA162,0,189,22,30,201,8,208,20,168,189,0,30,201,8,16,3,141,8,29,152,157,0
,30
1004 DATA169,32,157,22,30,224,255,240,4,232,76,137,3
1006 DATA162,0,189,22,31,201,8,208,20,168,189,0,31,201,8,16,3,141
1007 DATA8,29,152,157,0,31,169,32,157,22,31,224,200,240,4,232,76,174,3
1008 DATA162,0,169,32,157,22,30,232,224,22,240,3,76,211,3,96
1009 DATA169,126,141,4,28,169,36,141,5,28,169,36,141,6,28,169,24,141,7,28,96
1010 DATA169,102,141,4,28,169,60,141,5,28,169,66,141,6,28,169,129,141,7,28,96
1100 DATA60,126,219,255,102,60,66,129,66,36,24,126,165,126,24,
1101 DATA128,128,231,159,252,252,191,136,,126,255,255,66,66,255,255,1,1,193,249,
63,63
1102 DATA255,,,,,,,,,,,,,,,,,,,,,,,,,,,,8,8,28,28,28,28,20,54,,24,126,165
1104 DATA255,165,127,36,66,36,60,24,126,255,231,,,24,126,165,255,165,254,36
1105 DATA255,165,255,165,255,165,255,165,4,14,14,31,31,31,4,4,255,255,255,255,25
5,255
1106 DATA255,255,,60,126,126,126,126,126,126,102,102,102,102,102,102,102,102
1107 DATA,,,126,60,60,60,60,60,60,60,126,126,126,255,,,,,,52,126,255

READY.
```

Listing »Croussaider« (Schluß)

Mehr über SYS

Der SYS-Befehl beim C 64 und VC 20 leistet wesentlich mehr, als das Commodore-Handbuch zugeben will. Er bietet zum Beispiel eine einfache Möglichkeit, Parameter an Maschinenprogramme zu übergeben.



Bezeichnung	hex.-Adr.	dez.
Akkumulator	030C	780
X-Register	030D	781
Y-Register	030E	782
Proz.-Status	030F	783

Bild 1. Die Schnittstelle zu den Prozessor-Registern

```
1 REM SYS-DEMU
2 REM
3 REM CURSORPOSITION SETZEN/LESEN
4 REM
5 REM
10 A=780:REM AKKU
20 X=781:REM X-REGISTER
30 Y=782:REM Y-REGISTER
40 F=783:REM FLAG-REGISTER
50 UP=65520:REM ADRESSE VON PLOT
60 PRINT CHR$(147);"ZEILE 10, SPALTE 5"
70 :
100 REM CARRY-BIT LOESCHEN
105 REM ALSO CURSORPOS. SETZEN
110 POKE F,PEEK(F) AND 254
120 POKE X,10:REM ZEILE 10
130 POKE Y,5:REM SPALTE 5
140 SYS UP:REM AUFRUF CURSOR PLOT
150 PRINT"** CURSOR GESETZT";
190 :
200 REM CARRY SETZEN
205 REM ALSO CURSORPOS. LESEN
210 POKE F,PEEK(F) OR 1
220 SYS UP:REM AUFRUF PLOT
230 PRINT:PRINT:PRINT"CURSORPOSITION WAR:"
240 PRINT"ZEILE:";PEEK(X)
250 PRINT"SPALTE:";PEEK(Y)
260 PRINT:END
READY.
```

Bild 3. Listing Cursorposition setzen/lesen

N V – B D I Z C

- 2^0 = 1 = CARRY-Flag
- 2^1 = 2 = ZERO-Flag
- 2^2 = 4 = INTERRUPT-Flag
- 2^3 = 8 = DECIMAL-Flag
- 2^4 = 16 = BREAK-Flag
- 2^5 = 32 = unbenutzt
- 2^6 = 64 = OVERFLOW-Flag
- 2^7 = 128 = NEGATIVE-Flag

Bild 2. Das Flag-Register beim 6502

Der SYS-Befehl hat folgendes Format:
SYS <Adreßausdruck>
[,<Parameterausdrücke>]

SYS ruft das Maschinensprachprogramm auf, das bei »Adreßausdruck« logisch beginnt. »Adreßausdruck« steht für eine RAM-Adresse im Bereich von 0 bis 65535.

Die wahlweise anzugebenden Übergabeparameter werden nicht von SYS bearbeitet, vielmehr müssen diese Angaben in geeigneter Weise vom aufgerufenen Maschinenprogramm ausgewertet werden. Hierzu sind natürlich genauere Kenntnisse in Assembler-Programmierung erforderlich.

Es kann jedoch eine andere Übergabeform gewählt werden, auch wenn dies im VC 20-Programmierhandbuch verschwiegen und im Handbuch das Gegenteil behauptet wird.

Diese Form der Parameter-Übergabe besteht darin, Akkumulator, X- und Y-Register sowie den Prozessorstatus vorzugeben.

Wie soll das vom Basic aus geschehen? Wenn nicht direkt, dann eben über die Speicherstellen, die SYS als oben genannte Register aufnimmt, bevor ins Maschinenprogramm verzweigt wird, und in die SYS nach Rückkehr (RTS) aus dem Maschinenprogramm die aktuellen Registerinhalte ablegt.

Es gibt also eine Kommunikationsmöglichkeit mit dem Maschinenprogramm vor und nach der Abarbeitung, sie muß nur genutzt werden.

Die vier Speicherbytes nach Bild 1 sind die Schnittstellen zu den Prozessor-Registern.

Der Prozessor-Status wird dabei durch das Flag-Register dargestellt. Die Bedeutung der einzelnen Bits im Flag-Register geht aus Bild 2 hervor.

Hier nun eine Anwendung der Kommunikation mit Maschinen-Unterprogrammen.

Die Betriebssystem-Routine »Plot« soll genutzt werden, um den Cursor auf eine bestimmte Position zu setzen, ab der dann eine Ein-/Ausgabe erfolgen kann, beziehungsweise es soll abgefragt werden, wo sich der Cursor gerade befindet, wo also die nächste Ein-/Ausgabe erfolgen würde.

Bild 3 zeigt das kleine Beispiel-Listing. In den Zeilen 100 bis 150 wird der Cursor auf Zeile 10, Spalte 5 gesetzt. Anschließend wird zur Demonstration ein kleiner Text ausgedruckt und in den Zeilen 200 bis 260 die aktuelle Cursorposition gelesen.

(Rolf Zweifel)

Kopierprogramm für relative Files

Nicht jedes Kopierprogramm ist in der Lage, relative Dateien zu kopieren. Diese Arbeit übernimmt das folgende Programm. Aus Geschwindigkeitsgründen wurde es vollständig in Maschinensprache geschrieben.

Nach dem Starten des Programms erfolgen zunächst einige Abfragen:

1. Sind alle Erweiterungen ausgeschaltet? Damit ist zum Beispiel DOS 5.1 oder Simons Basic gemeint. Wird die Frage mit ja beantwortet, so steht der gesamte freie RAM (zirka 60 KByte) als Puffer zum Kopieren zur Verfügung. Andernfalls wird die Pufferobergrenze auf $ 8000 (32768) gesetzt, so daß zirka 28 KByte zur Verfügung stehen. In beiden Fällen können jedoch beliebig große Files kopiert werden.
2. Angabe der Gerätenummer und der Drivenummer von Ausgangslaufwerk und Ziellaufwerk. Zulässig sind die Gerätenummern 8 und 9 sowie die Drivenummern 0 und 1, bei anderen Nummern erscheint eine Fehlermeldung. Es wird die Gerätenummer 8 und die Drivenummer 0 auf dem Bildschirm vorgegeben, so daß bei Verwendung eines VC-1541-Laufwerks nur vier mal RETURN gedrückt werden muß.
3. Angabe des Filenamen der zu kopierenden relativen Datei. Beim Filenamen für das neue File wird der alte Filenamen vorgegeben, er kann natürlich mit den üblichen Tasten geändert werden.
4. Eingabe der maximalen Satzlänge des neuen Files. (Zulässig ist 1 bis 254). Sie muß nicht notwendigerweise mit der Satzlänge des alten Files übereinstimmen, sollte aber so lang sein wie der längste Satz im alten File. Andernfalls kommt es zu Da-

```
0 REM LADEPROGRAMM FUER RELATIV-KOPIERER
1 REM ERZEUGT MASCHINENPROGRAMM AUF DISK
ETTE
2 REM BERNWARD BRETTHAUER
3 REM BAURAT-GERBER-STR. 22
4 REM 3400 GOETTINGEN
5 REM TEL. 0551/58484
6 REM
10 REM TEST AUF  RICHTIGE PRUEFSUMMEN
20 READA:IF A>1000 THEN 100
30 IF A>=0 THEN S=S+A:GOTO20
35 B=B+1
40 IF S<>-A THEN PRINT"PRUEFSUMMENFEHLER
 IN BLOCK "B:F=1:S=0:GOTO20
50 PRINT"BLOCK "B"OK !":S=0:GOTO20
99 REM ERZEUGUNG DES MASCHINENPROGRAMMS
100 IF F THEN PRINT"DATAS FEHLERHAFT!":E
ND
110 PRINT"DISKETTE EINLEGEN UND <@> DRUE
CKEN"
120 GETA$:IF A$<>"@" THEN 120
130 PRINT"PROGRAMM WIRD ERZEUGT !"
140 OPEN2,8,2,"REL KOPIERER,P,W":RESTORE
150 PRINT#2,CHR$(1)CHR$(8);:REM LADEADRE
SSE
160 READ A:IF A<0 THEN160
170 IF A<1000 THEN PRINT#2,CHR$(A);:GOTO
160
180 CLOSE2:END
1000 DATA11,8,10,0,158,50,48,54,49,0,0,0
,165,1,133,57,169,255,141,199,2,169
1010 DATA126,160,12,32,252,11,32,228,255
,201,74,240,9,201,78,208,245,169,127
1020 DATA141,199,2,169,178,160,12,32,174
,11,141,192,2,120,173,20,3,201,13
1030 DATA240,3,141,200,2,173,21,3,201,12
,240,3,141,201,2,169,13,141,20,3,169
1040 DATA12,141,21,3,88,169,12,160,13,32
,222,11,141,52,3,169,21,160,13,32
1050 DATA174,11,141,194,2,169,39,160,13,
32,222,11,141,134,3,169,58,141,53
1051 DATA-11722:REM PRUEFSUMME BLOCK 1
1060 DATA3,141,135,3,169,48,160,13,32,25
2,11,160,2,32,207,255,201,13,240,6
1070 DATA153,52,3,200,208,243,192,2,240,
239,169,44,153,52,3,200,169,76,153
1080 DATA52,3,200,140,193,2,169,79,160,1
3,32,252,11,160,4,204,193,2,240,9
1090 DATA185,50,3,32,12,225,200,208,242,
136,136,136,136,169,157,32,12,225
1100 DATA136,208,250,160,2,32,207,255,20
1,13,240,6,153,134,3,200,208,243,192
1101 DATA-11891:REM PRUEFSUMME BLOCK 2
1110 DATA2,240,239,169,44,153,134,3,200,
169,76,153,134,3,200,169,44,153,134
1120 DATA3,200,140,195,2,169,110,160,13,
32,252,11,169,0,133,2,32,207,255,201
1130 DATA13,240,24,56,233,48,48,29,201,1
0,176,25,160,10,24,101,2,176,18,136
1140 DATA208,249,133,2,76,252,8,165,2,24
0,11,201,255,208,17,240,5,169,0,32
1150 DATA242,11,169,224,160,13,32,252,11
,76,241,8,172,195,2,153,134,3,238
1151 DATA-11244:REM PRUEFSUMME  BLOCK 3
1160 DATA195,2,169,0,133,2,133,66,133,68
,169,1,133,65,133,67,169,1,160,14
1170 DATA32,252,11,173,141,2,201,1,208,2
49,169,0,141,4,212,32,143,9,32,54
1180 DATA11,169,56,160,14,32,252,11,173,
141,2,201,1,208,249,169,0,141,4,212
1190 DATA32,77,10,32,54,11,165,2,240,202
```

Listing. Kopieren von relativen Files

```
,169,108,160,14,32,252,11,169,179
1200 DATA160,12,76,49,8,32,64,12,169,2,1
74,192,2,160,2,32,186,255,173,193
1201 DATA-9957:REM PRUEFSUMME BLOCK 4
1210 DATA2,162,52,160,3,32,189,255,32,19
2,255,32,80,11,240,8,104,104,32,53
1220 DATA12,76,136,9,165,64,205,199,2,20
8,4,32,53,12,96,162,15,32,201,255
1230 DATA169,80,32,12,225,169,2,32,12,22
5,165,65,32,12,225,165,66,32,12,225
1240 DATA32,204,255,32,80,11,240,8,169,2
55,133,2,32,53,12,96,169,32,32,12
1250 DATA225,165,66,166,65,32,205,189,16
9,13,32,12,225,169,145,32,12,225,162
1251 DATA-10061:REM PRUEFSUMME BLOCK 5
1260 DATA2,32,198,255,169,0,133,144,160,
0,200,32,207,255,162,52,120,134,1
1270 DATA145,63,166,57,134,1,88,165,144,
240,236,200,162,52,120,134,1,152,160
1280 DATA0,145,63,166,57,134,1,88,24,101
,63,133,63,169,0,101,64,133,64,32
1290 DATA204,255,169,1,24,101,65,133,65,
169,0,101,66,133,66,76,182,9,32,64
1300 DATA12,169,2,174,194,2,160,2,32,186
,255,173,195,2,162,134,160,3,32,189
1301 DATA-10430:REM PRUEFSUMME BLOCK 6
1310 DATA255,32,192,255,32,80,11,240,3,7
6,174,9,162,15,32,201,255,169,80,32
1320 DATA12,225,169,2,32,12,225,165,65,5
6,233,1,8,32,12,225,165,66.40,233
1330 DATA0,32,12,225,32,204,255,32,80,11
,162,2,32,201,255,169,255,32,12,225
1340 DATA32,204,255,32,80,11,240,3,76,17
4,9,165,68,197,66,208,10,165,67,197
1350 DATA65,208,4,32,53,12,96,162,15,32,
201,255,169,80,32,12,225,169,2,32
1351 DATA-10626:REM PRUEFSUMME BLOCK 7
1360 DATA12,225,165,67,32,12,225,165,68,
32,12,225,32,204,255,169,32,32,12
1370 DATA225,165,68,166,67,32,205,189,16
9,13,32,12,225,169,145,32,12,225,160
1380 DATA0,162,52,120,134,1,177,63,166,5
7,134,1,88,133,69,162,2,32,201,255
1390 DATA160.1,162,52,120,134,1,177,63,1
66,57,134,1,88,32,12,225,200,196,69
1400 DATA208,236,24,152,101,63,133,63,16
9,0,101,64,133,64,32,204,255,169,1
1401 DATA-10518:REM PRUEFSUMME BLOCK 8
1410 DATA24,101,67,133,67,169,0,101,68,1
33,68,76,170,10,169,40,141,1,212,169
1420 DATA15,141,5,212,169,0,141,6,212,16
9.15,141,24,212,169,17,141,4,212,96
1430 DATA162,15,32,198,255,32,207,255,14
1,197,2,201,48,240.53,32,207,255,141
1440 DATA198,2,201,48,208,16,173,197,2,2
01,53,208,9,32,207,255,201,13,208
1450 DATA249,240,27,169,13,32,12,225,173
,197,2,32,12,225,173,198,2,32,12,225
1451 DATA-11375:REM PRUEFSUMME BLOCK 9
1460 DATA32,207,255,32,12,225,201,13,208
,246,32,204,255,173,197,2,201,48,96
1470 DATA120,173,200,2,141,20,3,173,201,
2,141,21,3,88,108,2,160,32,252,11
1480 DATA32,207,255,72,169,0,141,4,212,3
2,242,11,104,201,81,240,219,201,56
1490 DATA240,20,201,57,240,16,104,104,16
9.142,160.13,32.252.11.169.251.160
1500 DATA12,76,49,8,56,233,48,96,32,252,
11,32,207,255,72,32,242,11,104,201
1501 DATA-11341:REM PRUEFSUMME BLOCK 10
1510 DATA48,240,4,201,49,208,217,96,201,
```

Listing. Kopieren von relativen Files (Fortsetzung)

tenverlust. Aus Geschwindigkeitsgründen wird übrigens der Fehlerkanal während des Kopierens nicht abgefragt, so daß der Fehler »overflow in record« nicht erkannt wird!

Nachdem alle Eingaben erledigt sind, beginnt das Kopieren. Das Programm gibt jeweils auf dem Bildschirm an, welche Diskette einzulegen ist, bei Diskettenwechsel und am Programmende wird zusätzlich ein Tonsignal erzeugt. Ist der Diskettenwechsel durchgeführt, so muß die SHIFT-Taste gedrückt werden, damit es weitergeht. Beim Arbeiten mit zwei Laufwerken oder einem Doppellaufwerk sowie beim Kopieren auf die gleiche Diskette kann die SHIFT-LOCK-Taste eingerastet werden, sobald alle benötigten Disketten im richtigen Laufwerk liegen. Es wird dann ohne Pause kopiert.

Ist das Kopieren beendet, startet das Programm von vorn. Es kann abgebrochen werden, indem als Gerätenummer »q« eingegeben wird. Der Rechner meldet sich dann wieder mit »READY« im Basic-Modus. Übrigens ist während des gesamten Kopierens (ab Eingabe der ersten Gerätenummer) die Stoptaste funktionsfähig. Wird sie gedrückt, so werden die offenen Files geschlossen und das Programm startet von vorn. Da während des Kopierens der augenblicklich kopierte Satz auf dem Bildschirm angezeigt wird, ist es damit auch möglich, ein File nur teilweise zu kopieren. Es ist normal, daß beim Schreiben die erste Satznummer erst nach längerer Zeit erscheint, da die Floppy das File zuerst einmal einrichten muß.

Folgende Fehlermeldungen werden vom Programm ausgegeben: ?? falsches Laufwerk oder Drive ??: Gerätenummer ungleich 8 oder 9 beziehungsweise Drivenummer ungleich 0 oder 1
?? Unzulässige Satzlänge ??: Satzlänge größer 254 oder kleiner 1
?? Gerät nicht bereit ??: Das angesprochene Laufwerk ist nicht eingeschaltet oder angeschlossen.
Abbruch !: Die Stoptaste wurde gedrückt.

```
13,240,251,32,207,255,76,242,11,133
1520 DATA34,132,35,160,0,177,34,240,6,32
,12,225,200,208,246,96,32,225,255
1530 DATA240,3,76,49,234,169,49,141,20,3
,169,234,141,21,3,88,169,0,141,4,212
1540 DATA32,53,12,162,248,154,169,210,16
0,13,32,252,11,76,211,11,169,2,32
1550 DATA195,255,169,15,32,195,255,96,16
9,15,174,192,2,160,15,32,186,255,169
1551 DATA-11829:REM PRUEFSUMME BLOCK 11
1560 DATA0,32,189,255,32,192,255,162,15,
32,201,255,176,24,169,73,32,12,225
1570 DATA32,204,255,169,148,133,63,169,1
4,133,64,169,136,160,14,32,252,11
1580 DATA96,32,204,255,169,181,160,13,32
,252,11,76,21,12,8,14,147,13,83,73
1590 DATA78,68,32,65,76,76,69,32,197,82,
87,69,73,84,69,82,85,78,71,69,78,32
1600 DATA65,85,83,45,13,71,69,83,67,72,6
5,76,84,69,84,32,40,74,47,78,41,63
1601 DATA-9399:REM PRUEFSUMME BLOCK 12
1610 DATA13,0,147,13,203,79,80,73,69,82,
69,78,32,86,79,78,32,82,69,76,65,84
1620 DATA73,86,69,78,32,198,73,76,69,83,
13,13,40,67,41,32,194,69,82,78,87
1630 DATA65,82,68,32,194,82,69,84,84,72,
65,85,69,82,13,13,32,32,32,32,32
1640 DATA32,32,32,32,49,57,56,52,13,13,8
6,79,78,32,32,204,65,85,70,87,69,82
1650 DATA75,32,56,157,0,196,82,73,86,69,
32,48,157,0,13,78,65,67,72,32,204
1651 DATA-7405:REM PRUEFSUMME BLOCK 13
1660 DATA65,85,70,87,69,82,75,32,56,157,
0,196,82,73,86,69,32,48,157,0,13,13
1670 DATA198,73,76,69,78,65,77,69,78,32,
68,69,83,13,65,76,84,69,78,32,198
1680 DATA73,76,69,83,32,63,32,0,13,13,19
8,73,76,69,78,65,77,69,78,32,68,69
1690 DATA83,13,78,69,85,69,78,32,198,73,
76,69,83,32,63,32,0,13,13,211,65,84
1700 DATA90,76,65,69,78,71,69,32,68,69,8
3,32,79,69,85,69,78,32,198,73,76,69
1701 DATA-7751:REM PRUEFSUMME BLOCK 14
1710 DATA83,32,63,32,0,17,17,13,63,63,32
,70,65,76,83,67,72,69,83,32,204,65
1720 DATA85,70,87,69,82,75,32,79,68,69,8
2,32,196,82,73,86,69,32,63,63,13,0
1730 DATA13,13,63,63,32,199,69,82,65,69,
84,32,78,73,67,72,84,32,66,69,82,69
1740 DATA73,84,32,63,63,13,0,13,13,193,6
6,66,82,85,67,72,32,33,13,13,0,13
1750 DATA13,63,63,32,213,78,90,85,76,65,
69,83,83,73,71,69,32,211,65,84,90
1751 DATA-7081:REM PRUEFSUMME BLOCK 15
1760 DATA76,65,69,78,71,69,32,63,63,13,0
,13,13,194,73,84,84,69,32,207,82,71
1770 DATA73,78,65,76,68,73,83,75,69,84,8
4,69,32,69,73,78,76,69,71,69,78,13
1780 DATA85,78,68,32,211,200,201,198,212
,32,68,82,85,69,67,75,69,78,32,33
1790 DATA13,0,13,13,194,73,84,84,69,32,2
18,73,69.76,68,73,83,75,69,84,84,69
1800 DATA32,69,73,78,76,69,71,69,78,13,8
5,78,68,32,211,200,201,198,212,32
1801 DATA-8582:REM PRUEFSUMME BLOCK 16
1810 DATA68,82,85,69,67,75,69,78,32,33,1
3,0,13,13,13,196,65,83,32,198,73,76
1820 DATA69,32,73,83,84,32,75,79,80,73,6
9,82,84,32,33,13,13,0,147,211,65,84
1830 DATA90,32,206,82,46,13,13,0
1851 DATA-3428:REM PRUEFSUMME BLOCK 17
19000 DATA 10000:REM ENDEKENNZEICHEN
```

Listing. Kopieren von relativen Files (Schluß)

Zusätzlich werden Fehlermeldungen der Floppy ausgegeben, falls sie auftreten (Außer RECORD NOT PRESENT). FILE TOO LARGE bedeutet, daß kein Platz mehr auf der Diskette ist.

Arbeitsweise des Programms:

Die Abfrage der Stoptaste geschieht mit Hilfe der Interruptroutine des Rechners, die 60 mal pro Sekunde durchlaufen wird und automatisch die Tastatur abfragt. In diese Interruptroutine wird eine zusätzliche Routine eingebunden, welche die Stoptaste abfragt. Ist die Stoptaste gedrückt, so wird zuerst ein eventuelles Tonsignal ausgeschaltet und alle Files werden geschlossen. Außerdem wird der Stackpointer zurückgesetzt, da der Abbruch ja in jeder Unterprogrammebene erfolgen kann. Danach erfolgt ein Neustart. Als Zwischenspeicher für die Filenamen wird der Kassettenpuffer, als Arbeitsbereich der Speicher für Sprite 11 (ab Adresse 704) verwendet. Die Sätze des Files werden in kompakter, sequentieller Form im RAM des Rechners untergebracht. Es wird zunächst die aktuelle Satzlänge +1 gespeichert und danach die Bytes des Satzes. Dadurch nehmen kurze Sätze auch nur wenig Platz im RAM ein (während sie auf der Diskette den gesamten Platz entsprechend der maximalen Satzlänge des Files belegen).

Eingabe des Programms:

Der Basic-Lader bildet nach jeweils fünf DATA-Zeilen eine Prüfsumme, so daß nahezu alle Eingabefehler erkannt werden. Nicht erkannt werden vergessene Nullen und überschüssige Kommata. Wenn alle Prüfsummen korrekt sind, fordert der Lader zum Einlegen einer Diskette auf. Das Programmfile »rel kopierer« wird dann direkt auf Diskette erzeugt. Das erzeugte Programm kann dann wie ein Basicprogramm geladen, kopiert und mit RUN gestartet werden.

(Bernward Bretthauer)

Listschutz

Einen verblüffenden Listschutz für einzelne Zeilen erhält man, indem man an die eigentliche Programmzeile einen REM-Befehl anhängt und dahinter in Anführungszeichen eine Reihe reverser »T« gefolgt von einem Doppelpunkt und einem beliebigen Text schreibt.

Geben Sie doch einmal folgendes ein:

10 PRINT"BAUM":REM"(19 reverse T)":10 PRINT "BLUME"

Wenn Sie dieses kleine Programm starten, schreibt der Computer »Baum«, listen Sie aber das Programm, so sehen Sie nur die Zeile 10 PRINT"BLUME".

(Roger Limberg)

INPUT ohne Fragezeichen

Die Ausgabe eines Fragezeichens beim INPUT-Befehl kann durch Öffnen einer Tastaturdatei unterdrückt werden:

10 OPEN 1,0 : REM Tastaturdatei eröffnen
20 INPUT#1,a$: REM Einlesen von Tastatur ohne Fragezeichen
30 REM Nicht vergessen, die Datei mit CLOSE 1 wieder zu schließen

POKEs für den 64er und den VC 20

POKE 775,200	Listschutz ein
POKE 775,167	Listschutz aus
POKE 788,49	Run/Stop ein
POKE 788,52	Run/Stop aus
POKE 808,237	Run/Stop-Restore ein
POKE 808,225	Run/Stop-Restore aus
POKE 650,128	Dauerfunktion für alle Tasten
POKE 650,0	Dauerfunktion nur für Space und Cursortasten
POKE 650,64	Dauerfunktion aus für alle Tasten

Zwei Einzeiler

Zahlenkonvertierungen von Dezimal nach Hexadezimal braucht man recht häufig. Hier sind zwei Einzeiler zu diesem Thema:

— Hex X$ nach dezimal X:10 x=0:fori=1tolen(x$): x0=asc (mid$(x$,i,1)):x=16*x+x0-48+(x0>64)*7: next

— Dezimal X nach hex X$:10 x$="":fori=1to4:x0=x/16: x=x-int(x0)*16:x$=chr$(48+x-(x>9)*7+x$:x=x0:next

Tastatur statt Joystick

Beim Commodore 64 lassen sich alle Joystick-Funktionen auch über die Tastatur steuern. Hier eine Liste der entsprechenden Tasten:

Joystick Port 1

Feuer	=	SPACE
Links	=	CTRL
Rechts	=	»2«
Oben	=	»1«
Unten	=	»<«

Joystick Port 2

Feuer	=	CTRL + »J«
Links	=	CTRL + »D«
Rechts	=	CTRL + »G«
Oben	=	CTRL + CRSR RIGHT
Unten	=	CTRL + »A«

(Gunther Knöpfle)

Zeitlupe für den VC 20

Läßt man ein Programm auf dem Bildschirm auflisten, so läuft es in Sekundenschnelle durch. Mit dem Befehl POKE 37877,0 werden nun alle Funktionen des VC 20 extrem verlangsamt, und man kann sich so ein Programm in Ruhe ansehen. Durch Drücken einer beliebigen Taste wird das Listen noch weiter verlangsamt und mit der RUN/STOP-Taste so lange angehalten, wie man die Taste gedrückt hält.

Man kann den Befehl auch innerhalb eines Programms anwenden um, zum Beispiel während der Testphase, bestimmte Abschnitte sehr langsam und somit nachvollziehbar ablaufen zu lassen.

Mit POKE 37877,72 oder einfach durch gleichzeitiges Drücken von RUN/STOP und RESTORE wird wieder der Normalzustand hergestellt. (Johannes Conrad)

Vom Bildschirm auf Kassette/Diskette

Das kurze Maschinenprogramm dient dazu, den Bildschirminhalt beim Commodore 64 abzuspeichern und auch wieder einzuladen. Dies kann mit einem Recorder oder Diskettenlaufwerk geschehen. Dabei werden außer dem eigentlichen Bildschirmspeicher (der an eine beliebige Stelle verschoben sein kann und nicht ab Adresse 1024 liegen muß) auch das Color-RAM und die Tabelle der Doppelzeilenkennzeichnungen mitberücksichtigt. Das Abspeichern und Einladen erfolgt über einen Pufferbereich im »versteckten« RAM ab 40960, da die drei verschiedenen Speicherbereiche in einem Stück abgespeichert und außerdem die 1000 Farbnibbles zu 500 Bytes zusammengeschoben werden. Das erspart beim Arbeiten mit dem Recorder wertvolle Zeit.

Nach dem Eintippen des Basic-Loaders kann durch »RUN 16« geprüft werden, ob es richtig eingegeben wurde. Trotzdem sollte es vor dem Start abgespeichert werden, da auch durch Prüfsummen nicht alle Fehler erkannt werden können. Ausgegeben werden zwei Adressen für das Laden und Speichern. Das Programm kann an jede Stelle des Arbeitsspeichers geladen werden, in der vorliegenden Version lädt es sich an das Ende des Basic-Arbeitsspeichers. Es ist davon auszugehen, daß das Programm direkt nach dem Einschalten ohne andere Erweiterungen eingeladen und gestartet wird.

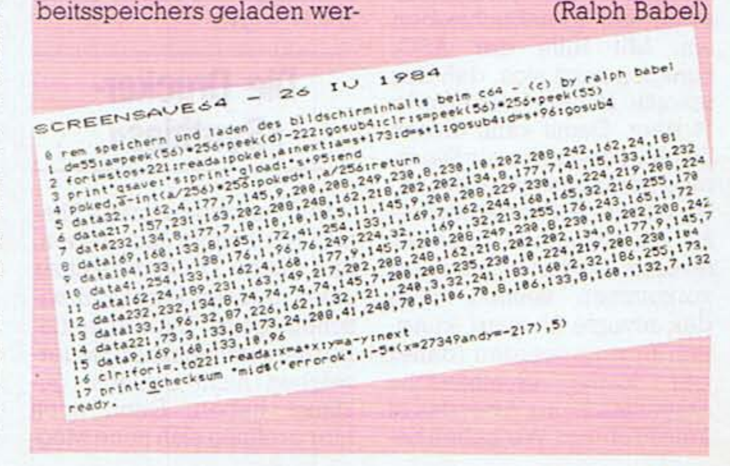
Soll ein Bildschirminhalt auf Diskette abgespeichert werden, so erfolgt dies durch
SYS 40738 »FILENAME«, 8

Beim Arbeiten mit Recorder kann (auch beim Einladen) die Gerätenummer entfallen; die Angabe einer Sekundäradresse ist nicht erlaubt. Durch das Abspeichern bedingte Betriebssystemmeldungen wie »PRESS RECORD & PLAY« sowie das Scrolling des Bildschirms sind unbedeutend und werden nicht berücksichtigt. Alle Bildschirmdaten werden zuvor in einen Zwischenspeicher übertragen.

Das Einladen solcherart abgespeicherter Bildschirminhalte geschieht mit
SYS 40833 »FILENAME«, 8

Zunächst werden die Daten in den Puffer von Adresse 40960 bis 42483 geladen und dann in MSB-Tabelle, Farbspeicher und die momentan gewählte Videomatrix übertragen.

(Ralph Babel)



```
SCREENSAVE64 - 26 IV 1984
0 rem speichern und laden des bildschirminhalts beim c64 - (c) by ralph babel
1 d=55:a=peek(56)*256+peek(d)-222:gosub4:clr:s=peek(56)*256+peek(55)
2 fori=stos+221:reada:pokei,a:next:a=s+173:d=s+1:gosub4:d=s+96:gosub4
3 print"qsave:"s:print"qload:"s+95:end
4 poked,a-int(a/256)*256:poked+1,a/256:return
5 data32,,162,4,177,7,145,9,200,208,249,230,8,230,10,202,208,242,162,24,181
6 data217,157,231,163,202,208,248,162,218,202,202,134,8,177,7,41,15,133,11,232
7 data232,134,8,177,7,10,10,10,10,5,11,145,9,200,208,229,230,10,224,219,208,224
8 data169,160,133,8,165,1,72,41,254,133,1,169,7,162,244,160,165,32,216,255,170
9 data104,133,1,138,176,1,96,76,249,224,32,,169,,32,213,255,176,243,165,1,72
10 data41,254,133,1,162,4,160,,177,9,145,7,200,208,249,230,8,230,10,202,208,242
11 data162,24,189,231,163,149,217,202,208,248,162,218,202,202,134,0,177,9,145,7
12 data232,232,134,8,74,74,74,74,145,7,200,208,235,230,10,224,219,208,230,104
13 data133,1,96,32,87,226,162,1,32,121,,240,3,32,241,183,160,2,32,186,255,173,
14 data221,73,3,133,8,173,24,208,41,240,70,8,106,70,8,106,133,8,160,,132,7,132
15 data9,169,160,133,10,96
16 clr:fori=.to221:reada:x=a+x:y=a-y:next
17 print"qchecksum "mid$("errorok",1-5*(x=27349andy=-217),5)
ready.
```

TEIL 2 SYNTHETISCHE

Benutzen Sie die synthetischen Steuerzeichen, um Ihre Druckerlistings übersichtlicher

Nachdem ich in der Ausgabe 5/84 des 64'er-Magazins die Erzeugung synthetischer Steuerzeichen und ihre Wirkung bei Bildschirmbetrieb dargestellt habe, soll jetzt von neuen Möglichkeiten für Druckeranwender die Rede sein. Dabei beziehe ich mich auf den Drucker VC 1515, mit dem die Steuerzeichen ausgetestet wurden. Es ist nicht auszuschließen, daß der eine oder andere Druckertyp unterschiedlich reagieren wird. Die wichtigsten Steuerfunktionen dürften jedoch auf allen Geräten die gleichen Reaktionen hervorrufen.

Bevor es nun zur Sache geht, sei mir — vor allem für neu hinzugekommene Leser — ein kurzer Abriß des vorangegangenen Artikels erlaubt:

Zur Cursor-Steuerung, für RVS ON/OFF oder zum Beispiel für Farbumschaltungen stehen dem VC 20- beziehungsweise C 64-Anwender zwei Möglichkeiten zur Verfügung. Erstens kann die CHR$-Funktion genutzt werden (Beispiel ?CHR$(19) bewirkt HOME) und zweitens erlauben reverse Steuerzeichen eine direkte und kurze Eingabe der Steuerbefehle (Beispiel: das reverse S veranlaßt ebenfalls HOME). Nun existieren jedoch einige CHR$-Codes, für die äquivalente Steuerzeichen nicht über zugehörige Tasten abgerufen werden können. So schaltet CHR$(14) beispielsweise auf Kleinbuchstaben um. Mit Hilfe der ASC-Funktion zeigt sich, daß das reverse N den CHR$-Code 14 trägt. Damit kann dieses Zeichen ebenfalls zur Steuerung herangezogen werden. Da außer den konventionellen Steuerzeichen keinerlei reverse Symbole in Strings vorkommen können, muß das reverse N quasi künstlich erzeugt werden (daher der Name »Synthetische Steuerzeichen«). Aber das ist kein Problem! Wir haben bereits ein Eingabeverfahren kennengelernt, das ich aber jetzt nicht wiederholen möchte. Statt dessen möchte ich Ihnen ein anderes Verfahren zeigen, das vielleicht ein wenig übersichtlicher ist, als das im vorigen Heft beschriebene.

Geben Sie die Programmzeile, die ein im String stehendes Steuerzeichen erhalten soll, wie gewohnt ein. Reservieren Sie dabei jedoch mittels Space die Stelle im Textstring, wo später das synthetische Steuerzeichen stehen wird. Schließen Sie die Eingabe der Zeile mit Betätigung der Return-Taste ab. Nun können sie ohne Schwierigkeiten den Cursor auf die bewußte Stelle bewegen, durch gleichzeitiges Drücken von CTRL und RVS ON in den Revers-Mode schalten (es erscheint kein reverses R!) und schließlich den freigehaltenen Platz mit dem entsprechenden Reversezeichen belegen (in unserem Beispiel also mit dem reversen N). Die so ergänzte Programmzeile wird jetzt durch erneutes Drücken der Return-Taste verlassen. Fertig. Auf diese Weise lassen sich sowohl die altbekannten als auch die neuen Steuerzeichen leicht erzeugen und entsprechend ins Programm einfügen. In den weiteren Ausführungen werde ich die synthetischen Steuerzeichen vereinfachend in geschweiften Klammern darstellen (zum Beispiel reverses N = {N}).

Die Drucker-»Synthies«

Nun zum eigentlichen Thema: Ich könnte fast wetten, daß einige Druckerbesitzer unter unseren Lesern inzwischen beim Experimentieren mit synthetischen Steuerzeichen nicht schlecht gestaunt haben. Denn auch hier eröffnen sich neue Möglichkeiten, an die bislang nicht zu denken war. So sind zwar keine spektakulären Effekte in laufenden Programmen zu erzielen, dafür jedoch hat man erstmals die Möglichkeit, Druckerlistings zu manipulieren. Zuvor möchte ich Ihnen jedoch eine Liste der Steuerzeichen geben. Es ist zu beachten, daß die mit *) gekennzeichneten Steuerzeichen nicht zu den synthetischen zählen, da sich diese direkt über Tasten eingeben lassen. Sie wurden nur der Vollständigkeit halber mit in die Tabelle aufgenommen.

Steuerzeichen	CHR$-Code	Wirkung
{H}	8	Umschaltung auf Grafik-Modus
{J}	10	Zeilenvorschub
{M}	13	RETURN. Nachfolgende Zeichen werden nicht mehr gedruckt.
{N}	14	Umschaltung auf Breitschrift
{O}	15	Umschaltung auf normale Schriftbreite
{P}	16	Festlegung der Druckstartposition
*){Q}	17	Umschaltung auf Kleinbuchstaben
*){R}	18	RVS ON
{Z}	26	Zeichenwiederholung (Grafik)
{[}	27	Punktadresse für Druckstart
{SHIFT M}	141	SHIFT RETURN. Nachfolgende Zeichen werden in einer neuen Zeile gedruckt.
*){SHIFT Q}	145	Umschaltung auf Großbuchstaben
*){SHIFT R}	146	RVS OFF

Tabelle. Liste der Steuerzeichen

Nun zu den Erläuterungen derjenigen Zeichen, die sinnvolle Anwendungen erlauben.

Grafik im Listing

Das reverse H gestattet die Umschaltung des Druckers in den Grafik-Modus. Alle nachfolgenden Zeichen im String werden — sofern ihr jeweiliger CHR$-Code größer als 127 ist — als Grafikinformationen interpretiert. In diesem Modus bleibt der Drucker so lange, bis er mit {N} oder {O} auf Schriftbetrieb zurückgeschaltet wird. Betrachten Sie bitte das Beispiel 1.

```
TEST⇔LAUF


10 REM GRAPHIK-MODUS
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST⇔LAUF"
50 PRINT#4:CLOSE4

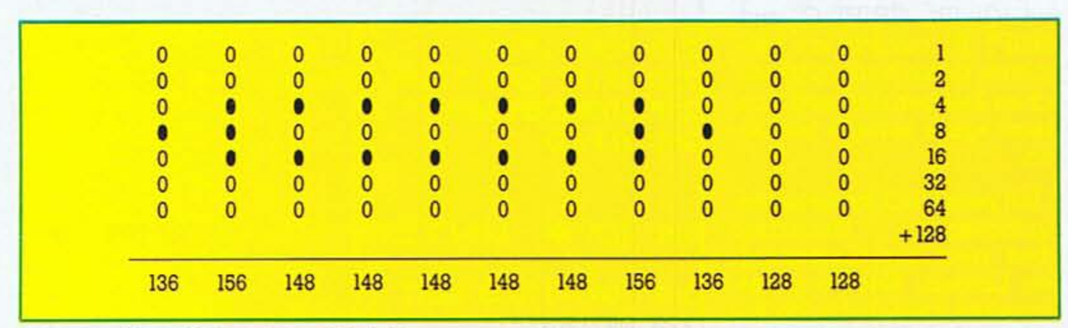
READY.
```

Beispiel 1

Zwischen »TEST« und »LAUF« befindet sich ein selbstdefiniertes Grafikzeichen, das sowohl im Programmlauf als auch im Listing ausgegeben wird. Auch wenn Sie es zunächst nicht glauben — Sie sehen ein Original-Listing und nicht etwa gePRINTete Textzeilen. Ich will Ihnen den Trick ver-

STEUERZEICHEN

zu machen. Es gibt interessante Möglichkeiten.



0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	2
0	●	●	●	●	●	●	●	0	0	0	4
●	●	0	0	0	0	0	●	●	0	0	8
0	●	●	●	●	●	●	●	0	0	0	16
0	0	0	0	0	0	0	0	0	0	0	32
0	0	0	0	0	0	0	0	0	0	0	64
											+128
136	156	148	148	148	148	148	156	136	128	128	

Bild 1. Vergrößertes Grafiksymbol aus Beispiel 1

raten. In einer Vergrößerung sieht das Grafiksymbol aus wie im Bild 1 gezeigt.

Die Umrechnung des dualen Bitmusters in Dezimalzahlen liefert diejenigen CHR$-Codes, durch die das Zeichen spaltenweise beim Drucken entsteht. Man muß also nur wiederum synthetische Zeichen finden, die die gesuchten CHR$-Codes tragen. Bei diesem Beispiel ergibt sich:

"TEST {H}	{SHIFT H}	{SHIFT MINUS}	{SHIFT T}	{SHIFT T}	{SHIFT T}	
8	136	156	148	148	148	
{SHIFT T}	{SHFIT T}	{SHIFT MINUS}	{SHIFT H}	{SHIFT*}	{SHIFT* }	{0}LAUF "
148	148	156	136	128	128	15

Mit dieser Folge von reversen Zeichen erzielen Sie also den Ausdruck, der im Beispiel 1 abgebildet ist.

Folgt dem {H} keine Grafikinformation, das heißt ist der CHR$-Code kleiner als 128, so kommt es zum Abbruch des Listings. Diese Tatsache kann man sich bewußt zunutze machen, wenn man ohne größeren Programmieraufwand einen Drucker-Listschutz in sein Programm einbauen will. Der Bildschirmbetrieb leidet darunter nicht, da {H} dort nur eine Verriegelung der Umschaltung zwischen Groß- und Kleinschrift bewirkt.

Zeilenvorschub

Die Beispiele 2.1 und 2.2 zeigen daß {J} einen Zeilenvorschub mit Rückwagenlauf verursacht. In 2.1 befindet sich das reverse J zwischen »TEST« und »LAUF«, bei 2.2 am Schluß des Strings. Da man das zweite Anführungszeichen eines Strings weglassen kann, sofern keine weiteren Basic-Befehle mehr in dieser Zeile folgen sollen, ergibt sich so die Möglichkeit, echte Leerzeilen mit {J} im Listing zu erzeugen

```
TEST
LAUF


10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TEST
LAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 2.1

```
TESTLAUF

10 REM ZEILENVORSCHUB
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF

50 PRINT#4:CLOSE4

READY.
```

Beispiel 2.2

Breitschrift — Schmalschrift

Während {N} beim Bildschirmbetrieb auf Kleinbuchstaben umschaltet, zeigt der Drucker eine andere Wirkung. Er wird veranlaßt, alle nachfolgenden Zeichen in doppelt breiter Schrift auszugeben. Es ist also bei der Anwendung von {N} Vorsicht geboten, da ein für den Bildschirm konzipiertes Programm beim Druckerlisting unerwünschte Steuerungen zur Folge haben kann. In diesen Fällen sollte man {N} wieder durch den konventionellen Befehl CHR$ ersetzen.

Wollen Sie jedoch die Breitschrift zur übersichtlichen Gestaltung eines Listings nutzen, so können Sie ruhig das {N} hier und da im Programm verstecken. Es bleibt demjenigen, der das Programm lesen soll, sowieso verborgen. Wenn nur eine Hervorhebung einzelner Worte, Befehle oder Zeilen gewünscht wird, muß die Breitschriftphase entsprechend mit {O} beendet werden. Im Beispiel 3 sehen Sie dieses Vorhaben realisiert. Hier bewirkt das {N} vor dem Wort »Test« die Umschaltung auf Breitschrift, während ein {O} vor »Lauf« diesen Modus beendet.

```
TESTLAUF


10 REM BREITSCHRIFT
20 REM IM LISTING
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 3

Druckstartposition

Die Druckstartpositionierung — vergleichbar mit TAB — wird normalerweise mit CHR$(16) und nachgestellter Startadresse vorgenommen. Also etwa:

PRINT #4,CHR$(16) "10TEST"

Diese Zeile druckt beginnend in Spalte 10 nur das Wort »TEST« aus. Die im String vorangestellte Zahl dient dabei als Startadresse. Das kürzere Äquivalent sieht nun wie folgt aus:

PRINT #4,"{P}10 TEST"

Doch Vorsicht! Obwohl diese Anweisung während

des Programmlaufs korrekt ausgeführt wird, verschluckt sie der Drucker im Listing vollständig. Das 4. Beispiel zeigt, daß von {P}10 zwischen »TEST« und »LAUF« nicht mehr zu bemerken ist.

```
TEST      LAUF


10 REM DRUCKSTARTPOSITION
15 REM IST IM LISTING
20 REM NICHT SICHTBAR
30 OPEN4,4
40 PRINT#4,"TESTLAUF"
50 PRINT#4:CLOSE4

READY.
```

Beispiel 4

Damit verbietet sich die Anwendung des synthetischen {P} bei Programmen, die zum Beispiel zur Veröffentlichung vom Drucker dokumentiert werden sollen.

Für die Zeichen {Z}Zeichenwiederholung und]{] (Punktadresse für Druckerstart) haben sich bislang noch keine sinnvollen Einsatzmöglichkeiten ergeben. Sie sollen daher jetzt nicht weiter diskutiert werden. Auch auf die Beschreibung von {Q}, {R}, {SHIFT Q} und {SHIFT R} möchte ich verzichten, da sie keine synthetischen Zeichen sind und sich damit »ganz normal« verhalten.

Ich persönlich benutze die synthetischen Steuerzeichen gern zur Gestaltung von Listings. Wie so etwas aussehen kann, habe ich Ihnen im abschließenden Beispiel 5 zusammengestellt.

Vielleicht versuchen Sie einmal, die versteckten und unsichtbaren Steuerzeichen herauszufinden.

(Jürgen Wagner)

```
100 REM DEMO ZUR OPTISCHEN STRUKTURIERUNG
110 REM     VON DRUCKER-LISTINGS MIT
120 REM"    SYNTHETISCHEN
130 REM"    STEUERZEICHEN


200 REM"*  HAUPTPROGRAMM  *

210 FORI=1TO64
220 PRINT"64'ER-MAGAZIN
230 GOSUB300:NEXT:"
240 END


300 REM"*  UNTERPROGRAMM  *

310 FORJ=1TO50:NEXT
320 RETURN:"


400 REM"ORIGINAL-LISTING!"

READY.
```

Beispiel 5

Autostart in Theorie und Praxis

Welcher Anwender hat sich nicht schon immer gewünscht, seine Programme »so einfach wie möglich« in den Computer zu bringen. Geräte der oberen Preis-/Leistungsklasse sind zu diesem Zweck mit einem Autostart-Mechanismus ausgerüstet. Dabei wird das Programm ohne weiteres Zutun nach dem Anschalten des Gerätes vom Massenspeicher (Floppy) in den Computer geladen. Hier soll jetzt die Realisierung eines solchen Autostarts auf einem C 64, beginnend mit der dazu nötigen Theorie, beschrieben werden.

Es gibt beim C 64 mehrere Möglichkeiten, einen Autostart herbeizuführen. Ich will Ihnen hier eine weniger bekannte Methode vorstellen:

Die Stack-Manipulation.

Der Stack belegt beim C 64 den Bereich $0100 bis $01ff (256 bis 511). Er wird unter anderem zum Ablegen von Rücksprungadressen benutzt. Nach Beendigung einer Routine sucht sich der Prozessor vom Stack zwei Bytes (nämlich die Rücksprungadresse in das Hauptprogramm) und springt die Adresse, die sich aus diesen beiden Bytes ergibt, an. Eigentlich ist daraus schon zu ersehen, was zu tun ist: Man müßte diese Rücksprungadresse so ändern, daß das Programm nicht an die eigentliche Rücksprungstelle springt, sondern auf eine eigene Routine. Jetzt jedoch tauchen schon die ersten Probleme auf: Wie soll der Stack geändert werden, ohne daß man dafür ein Extra-Programm braucht? Wie soll sich das Programm nach dem Ladevorgang automatisch starten?

Die Theorie

Auf beide Fragen gibt es eine Antwort: Da sich der Stack beim Ladevorgang ändern soll, ist es das einfachste, den (manipulierten) Stack einfach mit abzuspeichern! Damit hätten wir dann auch den von uns gewünschten Inhalt des Stacks geladen. Dann aber tauchen schon die nächsten Fragen auf: Woher wissen wir, aus welchem Stack-Bereich der Prozessor seine Rücksprungadresse holt? Die Antwort ist: Wir wissen es nicht! Unsere einzige Möglichkeit ist, den ganzen Stack mit der von uns gewünschten Rücksprungadresse zu belegen. Das Programm, auf das unsere Adresse zeigt (ein Maschinenprogramm), hängen wir direkt an den Stack an. Von dieser Startroutine wird nun das eigentliche Hauptprogramm, das wiederum hinter der Routine liegt, angesprungen. Die Reihenfolge der Programmteile und des benötigten Speichers ist in Bild 1 zusammengefaßt:

Startadresse des Programms auf Disk/Kas.: $0100	
$0100 — $01ff:	Stack, Lowbyte-1 und Highbyte der Startadresse unseres Startprogramms ($02)
$0200 — $0202:	Normaler Inhalt (unverändert)
$0203 — $0276:	Bereich für unser Startprogramm (technisch bedingt, immer gleich)
$0277 — $03ff:	Normaler Inhalt (unverändert)
$0400 — $07ff:	Bildschirmbereich, sollte Leerzeichen ($20) enthalten.
$0800:	Muß ein Null-Byte ($00) enthalten, damit der RUN-Befehl arbeiten kann.
$0801 — $xxxx:	Eigentliches Hauptprogramm, das vom Startprogramm angesprungen wird.

Bild 1. Programmadressen

Der Ladevorgang unseres Autostart-Programms muß mit »LOAD "xxx",8,1« erfolgen, damit das Programm nicht ab der Adresse $0801 (normaler Basic-Speicher), sondern ab $0100 geladen wird. Schauen wir uns nun noch einmal an, was im einzelnen beim Ladevorgang passiert:

1. Laden des Programms von Diskette oder Kassette mit »LOAD "xxx",8,1«.
2. Nach Beendigung des Ladevorgangs will sich der Prozessor vom Stack die Rücksprungadresse ins Basic holen, findet aber die (soeben geladene) Adresse auf unser eigenes Startprogramm und springt dies an.
3. Unser Startprogramm springt jetzt das eigentliche Hauptprogramm an, das irgendwo ab $0801 stehen sollte.

Nachdem also das Programm mit »LOAD "xxx",8,1« geladen wurde, startet es sich selbst sofort.

Zum Abschluß sei noch gesagt, daß einige Vorgänge zur besseren Verständlichkeit vereinfacht werden mußten. Bei eigenen Experimenten sei geraten, das abgedruckte Demo-Programm zu modifizieren. Außer einigen saftigen Abstürzen kann eigentlich nichts passieren.

Die Praxis

Das abgedruckte Demo-Programm läuft ohne Änderungen auf einem C 64 mit einer 1541-Floppy (Gerätenummer 8). Die Anpassung an andere Gerätenummern dürfte keine Schwierigkeit darstellen. Lediglich in den Zeilen 110, 112, 120 und 180 ist die 8 durch eine 9 zu ersetzen. Die Beschreibung der einzelnen Programmteile ist in Bild 2 zusammengefaßt.

Nun zum Programm selbst: Das Programm Autostart-Maker gibt dem Benutzer die Möglichkeit, ein beliebiges Programm mit einem Autostart zu versehen. Dabei muß folgende Bedingung erfüllt werden: Das Programm muß mindestens eine Basic-Zeile enthalten. Dies ist notwendig, da der vom Autostart-Maker generierte Autostart das Hauptprogramm mit dem RUN-Befehl startet. Zu diesem Zweck prüft der Autostart-Maker die Startadresse des gewünschten Programms und wirft eine Fehlermeldung aus, falls diese ungleich $0801 (Basic-Start) ist.

Wenn aber alles in Ordnung ist, so arbeitet das Programm eine Weile mit der Diskette, bis es eine Ende-Meldung ausgibt. Sollte ein Diskettenfehler auftreten, so macht sich das Programm optisch und akustisch bemerkbar. Danach hat der Benutzer die Wahl: das Programm zu beenden oder einen neuen Start zu versuchen.

Im Programm sind folgende Unterroutinen enthalten:
10000 Fehlerkanal lesen und auswerten
20000 Gong ausgeben (wird von Fehler-Routine aufgerufen)

In den Zeilen 30000- stehen die DATAs für das Startprogramm, das das eigentliche Hauptprogramm anspringen soll. Die Bedeutung einzelner Zeilen läßt sich auch den REM-Statements des Programms entnehmen.

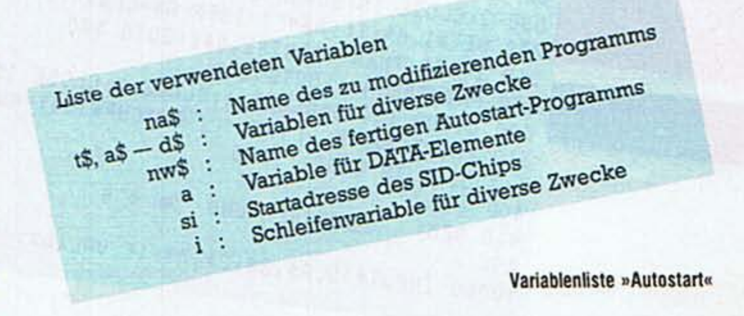
Ein mit diesem Autostart versehenes Programm ist auch in gewissem Sinne geschützt. Es läßt sich weder mit RUN/STOP noch mit RESTORE abbrechen. Wird extern ein RESET-Impuls erzeugt, so wird der ganze Speicher gelöscht, bevor in die normale RESET-Routine verzweigt wird. Diese Eigenschaften gehen auf den Aufbau unseres Startprogramms zurück.

Geladen wird das neue Programm mit »LOAD "name",8,1«. Der Name entspricht dem des Ursprungprogramms mit dem Zusatz »/a«. Noch ein Tip: Wenn Sie im Besitz eines Basic-Compilers sind, so sollten Sie den Autostart-Maker compilieren. Und nun viel Spaß mit dem Autostart-Maker. Wenden Sie ihn doch am besten gleich mal bei Ihrem Programm an.

(Andreas Wurf)

Zeilennr.	Funktion
0 — 50	Copyright & Ausgabe der Kopfzeile
60 — 61	Eingabe des Programmnamens und Kürzen auf 16 Zeichen
70 — 95	Warteschleife auf einen Tastendruck
100	Zusammensetzen des neuen Namens
110 — 165	Generieren des Programmteils, der den Autostart enthält, auf Diskette ($0100-$0800)
170— 300	Verbinden (Linken) des Autostart-Zusatzes und des eigentlichen Hauptprogramms
310	Ladeinstruktionen für das neu generierte, mit Autostart versehene Programm

Bild 2. Programmbeschreibung



Liste der verwendeten Variablen

na$:	Name des zu modifizierenden Programms
t$, a$ — d$:	Variablen für diverse Zwecke
nw$:	Name des fertigen Autostart-Programms
a :	Variable für DATA-Elemente
si :	Startadresse des SID-Chips
i :	Schleifenvariable für diverse Zwecke

Variablenliste »Autostart«

```
0 REM" *****************************
1 REM" **   ♠UTOSTART - \AKER ¬¬   **
2 REM" *    —OPYRIGHT (C) BY        *
3 REM" *      ♠NDREAS ◘URF          *
4 REM" *     2000  |AMBURG 73       *
5 REM" *    |ABENSTIEG  10 B        *
6 REM" *   |EL.: (040) 647 40 28    *
7 REM" **       XERSION —64        **
8 REM" *****************************
9 :
10 PRINT CHR$(9)+CHR$(14)+CHR$(8) "
UTOSTART - \AKER  ¬¬+             ■"
          COPYRIGHT (C) 1983 ♠NDREAS ◘URF
20 PRINT"
■"             —OMMODORE 64 - XERSION
30 PRINT"
■"
40 PRINT"(/OTE:  |OUR ‾ROGRAM MUST HAVE A |♠♥
-"
50 PRINT"♥TART, SUCH AS '10 ♥|♥ (XXXX)' !!)"
60 PRINT"NTER /AME OF ‾ROG.: ";:OPEN 1,0:INP
UT#1,NA$:CLOSE 1:PRINT
61 NA$=LEFT$(NA$,16):REM "┘UERZEN AUF 16 ♥TELLE
N
70 PRINT"NTER ‾ROG'S —ISK AND PRESS A KEY !"
80 FOR I=0 TO 20:GET A$:IF A$<>"" THEN 100
85 NEXT:PRINT"NTER ‾ROG'S —ISK AND PRESS A K
EY !■"
90 FOR I=0 TO 20:GET A$:IF A$<>"" THEN 100
95 NEXT:GOTO70
100 NW$=LEFT$(NA$,14)+"/A":REM" /AME DES NEUEN ‾
ROGRAMMS
110 OPEN 15,8,15:PRINT
112 OPEN 1,8,0,NA$:GOSUB 10000:CLOSE 1
120 OPEN 1,8,5,NW$+",P,W":PRINT"- |ENERATE ♠U
TOSTART":GOSUB 10000
130 PRINT#1,CHR$(0)+CHR$(1);:REM" ‾ROGRAMMSTART
:= $0100
140 FOR I=256 TO 514:PRINT#1,CHR$(2);:NEXT:REM"
LOW-1 / |IGHBYTE DES ♥TARTPGMS.
150 RESTORE:FOR I=515 TO 606:READ A:PRINT#1,CHR$
(A);:NEXT:REM" ♥TARTPROGRAMM
160 FOR I=607 TO 1023:PRINT#1,CHR$(PEEK(I));:NEX
T:REM" /ORMALER ¬NHALT
164 FOR I=1024 TO 2047:PRINT#1,CHR$(32);:NEXT:RE
M" |ILDSCHIRM MIT LEERZEICHEN
165 PRINT#1,CHR$(0);:GOSUB 10000:CLOSE 1
170 PRINT"- LINK TOGETHER BOTH —ILES"
180 OPEN 1,8,0,NA$:GOSUB 10000:OPEN 2,8,5,NW$+",
P,A":GOSUB 10000
190 GET#1,A$:A$=A$+CHR$(0):GET#1,B$:B$=B$+CHR$(0
):REM" ♥TARTADRESSE DES ‾GMS.
200 IF ASC(A$)=1 AND ASC(B$)=8 THEN 250:REM" |E
ST AUF |♠♥¬—‾ROGRAMM
210 PRINT"♥TART-♠DR. OF ♥OURCE-‾GM IS NOT UNIQ
UE"
220 PRINT"TO $0801. ¬ CAN'T USE IT.":CLOSE 1:CL
OSE 2:CLOSE 15:GOSUB 20000:END
250 GET#1,A$:IF A$="" THEN A$=CHR$(0)
260 IF ST THEN PRINT#2,A$;:GOTO 300
270 PRINT#2,A$;:GOTO 250
300 GOSUB 10000:CLOSE 1:CLOSE 2:CLOSE 15
310 PRINT"K. |YPE 'L‾♠-"CHR$(34)+NW$+CHR$(3
4)",8,1' TO LOAD.":END
400 :
410 REM"   _OUTINEN UND —♠| ♠'S
420 :
10000 INPUT#15,A$,B$,C$,D$:IF VAL(A$)=0 THEN RET
URN
10010 PRINT"-/‾ - ‾RROR# "A$": "B$" ON "C$",
"D$:GOSUB 20000
10020 PRINT"      UIT OR ESTART ?"
10030 GET T$:IF T$<>"Q" AND T$<>"R" THEN 10030
10040 CLOSE 1:CLOSE 2:CLOSE 15:IF T$="R" THEN RU
N
10050 END
20000 SI=54272:POKE SI+5,9:POKE SI+6,0:POKE SI+2
4,15
20010 POKE SI,30:POKE SI+1,30:POKE SI+4,17:FOR I
=0 TO 300:NEXT:POKE SI+4,0
20020 POKE SI,20:POKE SI+1,20:POKE SI+4,17:FOR I
=0 TO 600:NEXT:POKE SI+4,0
20030 POKE SI+24,0:RETURN
30000 DATA 169,52,162,193,141,20,3,142,24,3,162,
4,189,16,253,157,4,128,202
30010 DATA 16,247,169,57,162,2,141,0,128,141,2,1
28,142,1,128,142,3,128
30020 DATA 165,174,166,175,133,45,134,46,32,99,1
66,32,142,166,76,174,167
30030 DATA 169,0,162,8,133,158,134,159,160,0,169
,0,145,158,200,208,251,230
30040 DATA 159,165,159,201,208,208,239,169,0,162
,9,157,0,128,202,16,250
30050 DATA 76,226,252
READY.
```

Mit diesem Listing können Sie jedes Programm, das mindestens eine Basic-Zeile enthält, mit einem Autostart versehen, direkt nach dem Laden ausgeführt wird.







An seinem Foto können Sie schon erkennen, daß der Autor Bernd Pape eine Menge Witz und Phantasie besitzt. Auch in seinem »Lebenslauf« und seiner Spielbeschreibung kommt das ganz klar zum Ausdruck. Doch lesen sie selbst:

Anleitung für das Spielprogramm »Q Bernd«.

Was Sie hier erleben, ist der ebenso heldenhafte wie qualvolle Untergang des letzten Menschen der Erde: Q Bernd. Auf einem hyperwasserstoffbombenfesten Teil des Verwaltungsgebäudes der Antiwurgkommission ist es ihm gelungen, die Explosion zu überleben, die ein jähzorniger Erdkundelehrer in einem Wutausbruch durch seine übermächtigen Gehirnwellen erzeugt hat. Q Bernd hat sich auf dem folgenden langen Flug durch den Weltraum auf den leider nur sehr spärlich vorhandenen Sauerstoff einstellen können, was allerdings nicht ohne bleibende Schäden abgegangen ist, wie man an seiner dunklen Hautfarbe und seinen O-Beinen erkennen kann. Aber leider schwebt Q-Bernd mit seinem hyperwasserstoffbombenfesten Fetzen nicht allein durch die unendlichen Weiten des Weltraums. Als er ungefähr 765.854.863.654.392 Q-Berndsche Lichtmonate hinter sich gelegt hat, was nur dadurch möglich war, daß er sich jegliche Ernährung seines schon damals O-beinigen und sehr dunklen Körpers abgewöhnt hatte, näherte sich ihm ein

```
1 REM*********************
2 REM*                   *
3 REM* COPYRIGHT BY :    *
4 REM* BERND PAPE        *
5 REM* UNTERM SCHRICK 11 *
6 REM* 4630 BOCHUM 1     *
7 REM*                   *
8 REM*********************
10 PRINT"♥":POKE53281,0:POKE53280,2
20 PRINT:PRINT"E        R/ ■/| "
30 PRINT"     R/ N  E/ ■/| "
40 PRINT"    R/N        E/■/| "
50 PRINT"   R/N  ■E/     NR  E/■/| "
60 PRINT"  R/N  ■E/        NR  E/■/| "
70 FORI=0TO1:PRINT" NR  ■E|        NR  ■E
 | ":NEXTI
80 PRINT" NR  ■E|        NR  ■E|▒ ▒▒▒▒ ▒
 ▒▒▒▒ ▒▒ ▒ ▒▒▒▒ "
90 PRINT"□ NR  ■E|        NR  ■E|▒ ▒  ▒
▒   ▒  ▒ ▒▒ ▒ ▒  ▒"
100 PRINT"□ NR  ■E|        NR  ■E|▒ ▒▒▒
 ▒▒▒▒ ▒▒▒▒ ▒▒ ▒ ▒  ▒"
110 PRINT"□ NR  ■E|        NR  ■E|▒ ▒ ▒▒
 ▒▒▒▒ ▒▒   ▒ ▒▒ ▒  ▒"
120 PRINT"□ NR  ■E|        NR  ■E|▒ ▒  ▒
 ▒   ▒ ▒  ▒ ▒▒ ▒  ▒"
130 PRINT"□ NR  ■E|        NR  ■E|▒ ▒▒▒▒
 ▒▒▒▒ ▒  ▒ ▒ ▒▒ ▒▒▒▒ "
140 PRINT"□ NR  ■E|        NR  ■E| "
150 PRINT" NR  ■E|  R/ ■/| NR  ■E| "
160 PRINT" NR  E/■/| NR  E/ N  ■E/"
170 PRINT"  NR  E/■/| NR     ■E/"
180 PRINT"   NR  E/   N  E/■/| ♥(C) BY B
ERND PAPE 1984"
190 PRINT"    NR          E/■/| "
200 PRINT"        NR  ■E/  NR  E/
          ■/| "
210 PRINT"□              NR
         ■E/"
220 GOSUB1130
230 SI=54272:FORI=5TO19STEP7:POKESI+I,15
:POKESI+I+1,255:NEXTI:POKESI+24,15
240 POKE53280,2:PRINT"♥":FORI=1TO8:PRINT
:NEXT
```

Listing des Monats:



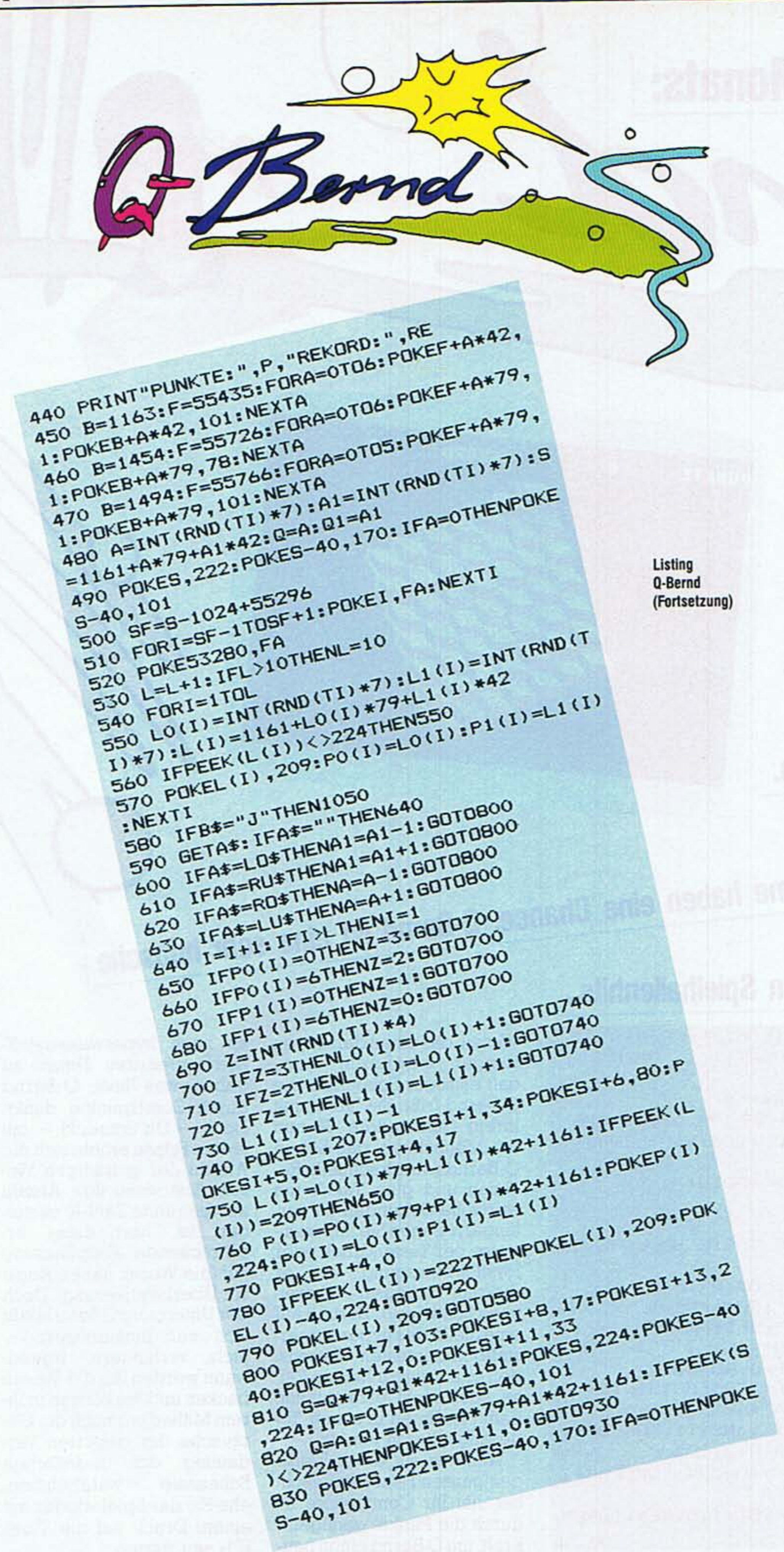
Listing des Monats sollen nicht nur sehr lange Programme werden.

Auch kleinere Programme haben eine Chance. Q-Bernd ist eine sehr hübsche Version eines bekannten Spielhallenhits.

```
250 PRINT"MOECHTEN SIE MIT":PRINT
260 PRINT"JOYSTICK(PORT1)      ODER
TASTATUR":PRINT
270 PRINT"SPIELEN ?":B$=""
280 IFPEEK(203)=22THENPOKE198,0:GOTO320
290 IFPEEK(203)=34ORPEEK(203)=60THENB$="
J":GOTO380
300 D=D+1:IFD>500THEND=0:GOTO10
310 GOTO280
320 PRINT""
330 PRINT:PRINT"WELCHE TASTE SOLL HEISS
EN :":PRINT
340 INPUT"-LINKS OBEN";LO$:PRINT
350 INPUT"-RECHTS OBEN";RO$:PRINT
360 INPUT"-LINKS UNTEN";LU$:PRINT
370 INPUT"-RECHTS UNTEN";RU$
380 PRINT"":L=0:FA=3:P=0:RU=48
390 B=1160:F=55432:FORA=0TO6:GOTO1030
400 FORA1=0TO6:H=A*79+A1*42
410 FORI=0TO2:POKEF+H+I,1:NEXTI:POKEB+H,
233:POKEB+H+1,224:POKEB+H+2,105
420 FORI=0TO1:POKEB+H+I+40,224:POKEF+H+I
+40,2:NEXTI:NEXTA1:NEXTA
430 FORI=4TO18STEP7:POKESI+I,0:NEXTI:RES
TORE
```

Wesen, das einen derartigen Hunger entwickeln kann, daß es sogar Scheusale wie Q-Bernd frißt. Sie haben mit Ihrem Commodore 64 nun die Möglichkeit, das Leben Q-Bernds durch schnelle Reaktion und gute Taktik um wertvolle Sekunden zu verlängern. Sagen Sie ihm wahlweise per Tastendruck oder Joystickbewegung, wie er diesem Scheusal ausweichen kann. Jeder Sprung auf eine Stelle des hyperwasserstoffbombenfesten Fetzens, den er auf seiner langen Reise noch nicht erreicht hat, dankt er Ihnen in seiner Todesangst durch Punkte.

Wenn er auf diese Weise den ganzen Fetzen erforscht hat, hat ihr Commodore 64 durch die Punkte genügend Kraft, um Q Bernd einen neuen noch hyperwasserstoffbombenfesteren Fetzen zu geben, was Ihnen Q Bernd durch Zusatzpunkte dankt. Doch — Oh Schreck! — mit jedem Fetzen erhöht sich die Anzahl der gefräßigen Wesen. Erst wenn ihre Anzahl auf die runde Zahl 10 gestiegen ist, hört diese erschreckende Fortpflanzung auf (die Wesen haben Angst vor Überbevölkerung). Doch der Untergang Q Bernds läßt sich nur hinauszögern — nicht verhindern. Irgendwann werden ihn die Wesen packen und Sie können in Ihrem Mitleid nur noch die Geräusche der gestörten Verdauung der widerlichen Scheusale wahrnehmen, ehe Sie das Spiel wieder mit einem Druck auf die Taste »F1« neu starten.



```
440 PRINT"PUNKTE:",P,"REKORD:",RE
450 B=1163:F=55435:FORA=0TO6:POKEF+A*42,
1:POKEB+A*42,101:NEXTA
460 B=1454:F=55726:FORA=0TO6:POKEF+A*79,
1:POKEB+A*79,78:NEXTA
470 B=1494:F=55766:FORA=0TO5:POKEF+A*79,
1:POKEB+A*79,101:NEXTA
480 A=INT(RND(TI)*7):A1=INT(RND(TI)*7):S
=1161+A*79+A1*42:Q=A:Q1=A1
490 POKES,222:POKES-40,170:IFA=0THENPOKE
S-40,101
500 SF=S-1024+55296
510 FORI=SF-1TOSF+1:POKEI,FA:NEXTI
520 POKE53280,FA
530 L=L+1:IFL>10THENL=10
540 FORI=1TOL
550 L0(I)=INT(RND(TI)*7):L1(I)=INT(RND(T
I)*7):L(I)=1161+L0(I)*79+L1(I)*42
560 IFPEEK(L(I))<>224THEN550
570 POKEL(I),209:P0(I)=L0(I):P1(I)=L1(I)
:NEXTI
580 IFB$="J"THEN1050
590 GETA$:IFA$=""THEN640
600 IFA$=LO$THENA1=A1-1:GOTO800
610 IFA$=RU$THENA1=A1+1:GOTO800
620 IFA$=RO$THENA=A-1:GOTO800
630 IFA$=LU$THENA=A+1:GOTO800
640 I=I+1:IFI>LTHENI=1
650 IFP0(I)=0THENZ=3:GOTO700
660 IFP0(I)=6THENZ=2:GOTO700
670 IFP1(I)=0THENZ=1:GOTO700
680 IFP1(I)=6THENZ=0:GOTO700
690 Z=INT(RND(TI)*4)
700 IFZ=3THENL0(I)=L0(I)+1:GOTO740
710 IFZ=2THENL0(I)=L0(I)-1:GOTO740
720 IFZ=1THENL1(I)=L1(I)+1:GOTO740
730 L1(I)=L1(I)-1
740 POKESI,207:POKESI+1,34:POKESI+6,80:P
OKESI+5,0:POKESI+4,17
750 L(I)=L0(I)*79+L1(I)*42+1161:IFPEEK(L
(I))=209THEN650
760 P(I)=P0(I)*79+P1(I)*42+1161:POKEP(I)
,224:P0(I)=L0(I):P1(I)=L1(I)
770 POKESI+4,0
780 IFPEEK(L(I))=222THENPOKEL(I),209:POK
EL(I)-40,224:GOTO920
790 POKEL(I),209:GOTO580
800 POKESI+7,103:POKESI+8,17:POKESI+13,2
40:POKESI+12,0:POKESI+11,33
810 S=Q*79+Q1*42+1161:POKES,224:POKES-40
,224:IFQ=0THENPOKES-40,101
820 Q=A:Q1=A1:S=A*79+A1*42+1161:IFPEEK(S
)<>224THENPOKESI+11,0:GOTO930
830 POKES,222:POKES-40,170:IFA=0THENPOKE
S-40,101
```

Listing Q-Bernd (Fortsetzung)

Und nun das Gleiche für Praktiker:

Steuern Sie ihr Männchen wahlweise mit Joystick oder Tastatur so, daß alle 49 Felder gefärbt werden, um in die nächste Phase zu gelangen. In jeder Phase bis zur Phase 10 erhöht sich die Anzahl der Figuren, deren Berührung mit Ihrem Männchen tödlich ist. Danach wird das Spiel nicht mehr schwieriger. Mit Druck auf die Taste »f1« können Sie das Spiel neu starten. (Bernd Pape)

Lebenslauf

Am 6.12.1968 wurde ich als angeblich gewolltes Kind meiner Eltern versehentlich im Ruhrgebiet geboren und wenig später in Stade, einer sehr schönen Stadt zwischen Hamburg und Cuxhaven auf den Namen Bernd getauft. Bernd Pape war also der Name, der von nun an meine Umwelt in Schrecken versetzen sollte. Ungefähr 6½ Jahre später gelang meinen Eltern ein kaltblütiges Attentat: Sie schulten mich in Bochum, wo ich auch jetzt noch meine Residenz habe, gegen meine schärfsten Proteste ein. Als ich nach 4 Jahren meinte, dieses Übel überstanden zu haben und ich gerade das Wort »arbeiten« aus meinem Vokabular streichen wollte, gelang es der Bürokratie, einen entscheidenden Sieg über mich zu erlangen: Ich wurde zum Gymnasiasten erklärt. All diese Mißhandlungen konnten mich aber nicht davon abhalten, meinem Protest durch ausgiebige Freizeitnutzung Nachdruck zu verleihen. Zunächst hatte ich nur die Möglichkeit, meine Eltern und die Nachbarschaft durch falsches, aber lautes Akkordeonspielen zu ärgern. Mit meinen ersten Elektronikkästen konnte ich dann aber mein Waffenlager erweitern: Ich erging mich in den herrlichsten und lautesten Sirenenschaltungen des 19. Jahrhunderts — oder des zwanzigsten? Geldgeschenke zur Konfirmation (weitere Spenden nehme ich gerne entgegen!) ermöglichten es mir, alle Leser dieser Zeitung aufzufordern, den Kampf um möglichst viel Freizeit nicht aufzugeben! Denn von diesem Geld holte

ich mir den Commodore 64, auf dem ich den »Q Bernd« unter besonders erschwerten Bedingungen geschrieben habe. Nach einer Lateinarbeit, deren Zensur nicht genau den Vorstellungen meiner Eltern entsprach, erhielt ich »Computer-Verbot«. Gott sei Dank sah ich dies schon beim Schreiben der Arbeit voraus, was dazu führte, daß ich meine Diskette mit dem schon angefangenen Programm vor der liebenden Umsicht meiner Eltern retten konnte. So entstand der Rest des Programms durch mein heimliches Programmieren in verschiedenen Computerläden Bochums. Hier ist es nun, das Programm, für das Bernd Pape seinen Hausfrieden geopfert hat: Q Bernd!

(Bernd Pape)

```
840 POKESI+11,0:SF=S-1024+55296:IF(PEEK(
SF)AND15)=1THEN860
850 GOTO640
860 FORW=SF-1TOSF+1:POKEW,FA:NEXTW:P=P+1
:IFP>RETHENRE=P
870 PRINT"██",P,,RE:IFP=RUTHEN890
880 GOTO640
890 FA=FA+1:IFFA>15THENFA=3
900 POKE53280,FA:RU=RU+48+10*L:A$="":P=P
+10*L:IFP>RETHENRE=P
910 PRINT"▙":GOTO390
920 IFA=0THENPOKES-40,101
930 FORI=0TO8:PRINT:NEXTI
940 FORI=180TO130STEP-1
950 POKESI+24,15:POKESI,207:POKESI+1,I:P
OKESI+6,143
960 POKESI+5,15::POKESI+4,17:NEXTI:POKES
I+4,0
970 POKESI,90:POKESI+1,4:POKESI+6,255:PO
KESI+5,15
980 POKESI,50:POKESI+1,17:POKESI+6,32:PO
KESI+5,0:POKESI+4,17
990 FORT=1TO180:NEXTT:POKESI+4,0
1000 PRINT"<<<<<<<<<  S P I E L - E N D E
 >>>>>>>>>>"
1010 PRINT:PRINT"█  F1 FUER EIN NEUES S
PIEL DRUECKEN !"
1020 POKE198,0:WAIT198,1:GOTO240
1030 READK:READKL:READKH
1040 POKESI+K,KL:POKESI+K+1,KH:POKESI+K+
4,17:GOTO400
1050 R=(255-PEEK(56321))AND15
1060 IFR=0THEN640
1070 IFR=4THENA1=A1-1:GOTO800
1080 IFR=8THENA1=A1+1:GOTO800
1090 IFR=1THENA=A-1:GOTO800
1100 IFR=2THENA=A+1:GOTO800
1110 GOTO640
1120 DATA0,180,8,7,247,10,7,10,13,7,103,
17,14,237,21,14,20,26,14,207,34
1130 POKE56,48:FORI=828TO870:READQ:POKEI,Q:NEXT
1140 FORI=1TO21:READA:NEXT
1150 DATA120,169,49,133,1,169,0,133,98,1
33,100,169,208,133,99,169,48,133,101
1160 DATA162,16,160,0,177,98,145,100,200
,208,249,230,99,230,101,202,208,242
1170 DATA169,55,133,1,88,96
1180 SYS828:POKE53272,(PEEK(53272)AND240
)OR12
1190 READA:IFA<0THENRESTORE:RETURN
1200 FORJ=0TO7:READB:POKE12288+A*8+J,B:N
EXT:GOTO1190
1210 DATA160,255,255,255,255,255,255,255
,255
1220 DATA170,255,255,231,195,195,129,0,0
1230 DATA222,129,195,219,189,219,24,255,
255
1240 DATA209,195,129,36,0,0,60,153,195,-
1
READY.
```

Listing Q-Bernd (Schluß)

Druckfehler-teufelchen



Die Redaktion hatte mir ausdrücklich den Kampf erklärt. Dennoch konnte ich wieder zweimal zuschlagen und Verwirrung stiften.

Schatzsucher, Seite 91 Ausgabe 5/84

»Nein, die Zeile 440 im Listing »Schatzsucher« fehlt nicht« wurde zur Standardantwort. Der Autor dieses Listings hatte anscheinend, bevor er den Beitrag einsandte, ein Renumber durchgeführt. Nachträglich wurde dann von ihm die Zeile 440 entfernt. Was in dieser Zeile stand, wird wohl immer ein Rätsel bleiben. Auf den Programmablauf hatte sie auf jeden Fall keinen Einfluß. Trotzdem bedanken wir uns bei den zirka eintausend Lesern, die uns im letzten Monat diesen »Fehler« meldeten.

Nun zu einigen wirklichen Fehlern. Wir möchten uns auch hier noch bei unseren Mitstreitern im Kampf gegen den Fehlerteufel bedanken.

Supergrafik ohne Erweiterungsmodul, Seite 81, Ausgabe 5/84

In Zeile 4030 muß »BC+1« durch »BC=BC+1« ersetzt werden, sowie in Zeile 4050 P = PEEK(P): durch P = PEEK(B):

Adreß- und Telefonregister, Seite 67, Ausgabe 5/84

Zeile 1060 muß lauten:
1060 FORI=0TOAN

ALLE TASTEN-, ZEICHEN- U

Das ist der dritte Teil einer Serie über die Abfragemethoden für Tasten und ihre ve und den C 64 gleichermaßen. Wo Unterschiede auftreten, sind di

Ich habe im zweiten Teil dieser Reihe versucht, Sie zum Experimentieren mit den ASCII-Codes anzuregen. Heute habe ich für Sie die vollständige Liste aller 255 Codewerte vorbereitet und zwar in einer Art, die sicher einiger Erklärungen bedarf.

ASCII-Codes, die von Commodore verschwiegen werden

Ich habe nämlich noch ein paar zusätzliche Überraschungen parat, auf die man nur durch Zufall kommt oder durch Studium des Betriebssystems oder aber, wenn man den Aufsatz von G. Urbanczyk in Computer persönlich vom 19.10.1983, Seite 76, gelesen hat.

Tippen Sie bitte das Programm Nummer 3 aus dem 64'er Mai-Heft, Seite 107, ein, nämlich eine der drei Versionen zur Tastaturpuffer-Abfrage. (Ich verwende unten die GET-Version).

```
310 PRINT CHR$(147)
320 GET A$
330 IF A$= " " THEN 320
340 PRINT ASC(A$)
350 GOTO 320
```

Auf beiden, VC 20 und C 64 erhalten Sie nach RUN 310 und Drücken der RETURN-Taste (natürlich) den ASCII-Code 82. Wenn Sie zuerst die CTRL-Taste drücken und halten und dann erst das R drücken, dürfte eigentlich nichts passieren, denn die CTRL-Taste gilt ja angeblich nur für die Farben. Ja, denkste! Wir erhalten nämlich die Zahl 18. Ein Blick in die ASCII-Tabelle zeigt uns für 18 die Funktion »REVERSE-ON«.

Versuchen Sie dasselbe mit CTRL und der ←-Taste. Wir erhalten die 6, und nicht 95, wie es eigentlich sein sollte.

Für den VC 20 ist das alles. Aber immerhin, wir haben sozusagen noch zwei zusätzliche Funktionstasten gefunden.

Beim C 64 aber geht es erst richtig los:

Der Versuch wird Ihnen zeigen, daß alle Buchstaben, von A bis Z, zusammen mit CTRL gedrückt, einen anderen ASCII-Wert, nämlich 1 bis 26, ergeben, als allein gedrückt.

Desweiteren biete ich Ihnen noch:

CTRL — ↑ = 30
CTRL — = = 31
CTRL — £ = 28
CTRL — : = 27
CTRL — ; = 29

Das heißt aber, daß einige ASCII-Codezahlen zwei Bedeutungen haben. Oder umgekehrt, zwei verschiedene Tasten (kombiniert) haben denselben ASCII-Code.

Schwierigkeiten dadurch, daß einige ASCII-Werte zwei Bedeutungen haben, gibt es deswegen nicht, weil die Kombination mit CTRL nicht PRINT-bar ist (PRINT CHR$(19) schickt immer den Cursor »home«, mit dem »S« passiert gar nichts).

Andersherum kann es allerdings vorkommen, daß eine Tastenabfrage, zum Beispiel
GET A$:IF A$ = CHR$(19) THEN ......
sowohl auf die Taste »HOME« als auch auf »CTRL-S« reagiert. Da ist sicher etwas Vorsicht angebracht. Aber ein Blick in meine ASCII-Tabelle zeigt Ihnen ja die Doppeldeutigkeiten.

An dieser Stelle erwarte ich eigentlich einige Einsprüche, wie: »Wozu das alles, die acht Funktionstasten, oder gar erweitert auf 32, reichen doch völlig aus!« Für den Haus- beziehungsweise Spielgebrauch ist das sicher richtig. Aber bei professioneller Software, welche benutzerfreundlich aufgebaut ist, kann es oft gar nicht genügend Funktionstasten — besonders solche, die eine optische Buchstabenbeziehung zu der Abfrage haben sollen — geben. Wenn in einem Programm gefragt wird, ob Sie »LOADen« oder »SAVEn« wollen, ist CTRL-L oder CTRL-S halt klarer, als f-1 oder f-3.

Ich finde es schade, daß diese großartige Möglichkeit nur auf dem C 64 gegeben ist. Hier zeigt sich deutlich, daß dieser Computer doch professioneller ist als der VC 20.

Die vollständige ASCII-Tabelle

So, jetzt können Sie meine ASCII-Tabelle erst richtig interpretieren (Tabelle 1).

Leere Kästchen haben keine Bedeutung für die betreffende Codezahl.

```
5 REM**********************************
6 REM***** SPIEL MIT FINKELN **********
7 REM**********************************
10 PRINT CHR$(147)
20 F=0
30 R=65
40 FOR T=1 TO 600:NEXT T
50 A=INT(RND(0)*7)+65
60 IF R>71 THEN 400
70 PRINT CHR$(A);
80 FOR T=1 TO 1000:NEXT T
100 GET A$
110 IF A$<>"[illegible]" THEN 300
120 PRINT"[illegible]";
200 IF A=R THEN F=F+1
210 GOTO 50
300 IF A<>R THEN F=F+1
310 R=R+1
320 GOTO 50
400 PRINT"[illegible]DAS SPIEL IST ZU ENDE";"[illegible]
"F"FEHLER"
```

Programm 2
Spiel mit »Finkeln« für C 64

ND STEUERCODES TEIL 3

chiedenen Codes. Alle Angaben gelten für den VC 20 Werte für den Commodore 64 in Klammern gesetzt.

Jeweils zwei Zeichen nebeneinander mit derselben Codezahl stellen die beiden Zeichensätze dar, in die mit C=SHIFT (Commodore-Taste) umgeschaltet werden kann. Wo nur ein Zeichen steht, ist es in beiden Zeichensätzen identisch.

Die Funktionen der Codezahlen 129 und 149 bis 155 gelten nur für den C 64. Interessant ist übrigens, daß die 4. und 7. Spalte identisch ist, ebenso die 6. und 8. Spalte (außer dem Zeichen für 255).

Ich möchte jetzt gern die Szene wechseln, ohne aber den ASCII-Code aus den Augen zu verlieren. Wir haben den ASCII-Code bisher verwendet, um Tasten abzufragen oder Funktionen auszuführen. PRINT CHR$(66) druckt zum Beispiel den Buchstaben B auf den Bildschirm.

Welche Methoden kennen Sie noch, mit denen das gleiche erzielt werden kann?

Die erste, die jeder aus dem Handbuch lernt, ist
PRINT "B".
Die komplizierteste ist:
POKE 7680,2: POKE 38400,7
(POKE 1024,2: POKE 55296,7)

Diese beiden Vorgehensweisen wollen wir uns näher anschauen und prüfen, ob wir sie in Analogie zu dem ASCII-Code für Tastenabfragen einsetzen können.

Es ist sicher viel bequemer, längere Buchstabenreihen oder gar Texte zwischen Gänsefüße gestellt einzutippen, als eine Serie von CHR$-Werten, ganz abgesehen vom erforderlichen Speicherplatz.

Nicht ganz so bequem ist der Gänsefuß-Modus bei Steuerzeichen, wie zum Beispiel Cursor-Bewegungen, besonders, wenn man diese herbeiführen will, aber statt dessen die reversen Darstellungen auf dem Bildschirm erzeugt.

Der Gänsefuß-Modus

Geben Sie es zu. Sie haben deswegen auch schon herzhaft geflucht. Auch jeder Redakteur bittet um Listings mit CHR$-Darstellung anstelle der reversen Zeichen, die bei der Druckwiedergabe oft zu Schwierigkeiten führen.

Jetzt wissen Sie, warum ich bei meinen Progrämmchen immer PRINT CHR$(147) statt PRINT" " verwende.

Genauso austauschbar wie bei PRINT ist der ASCII-Code mit dem Gänsefuß-Modus bei der Tastenabfrage.
Statt:
10 GET A$
20 IF A$< >CHR$(65) THEN 10
30 PRINT CHR$(88)
können wir schreiben:
10 GET A$
20 IF A$< >"A" THEN 10
30 PRINT "X"

Beide Programme sind gleichwertig. Nach RUN rührt sich gar nichts. Erst, wenn die A-Taste gedrückt wird (Zeile 20), druckt Zeile 30 den Buchstaben X.

In Zeile 20 können wir natürlich statt des A jeden beliebigen Buchstaben, Zahl oder Zeichen nehmen.

LISTen Sie einfach die 2. Version der drei Zeilen, fahren mit dem Cursor auf das A und verändern Sie es Ihren Wünschen entsprechend. Wie ich Sie einschätze, machen Sie das sicher auch mit den Funktionstasten.

Nein? Dann machen Sie es mal. Sie haben nämlich Pech, so geht es nicht. Aber es geht, wenn Sie sich mit Absicht in die Lage begeben, die wie vorhin beschrieben, Flüche auslöst. Fahren Sie mit dem Cursor auf den 1. Gänsefuß, tippen Sie ihn noch einmal ein und drücken Sie dann eine Funktionstaste. Siehe da, es erscheint ein reverses Zeichen. Mit RETURN wird es »fixiert«, nach RUN wird das X erst mit der verwendeten Funktionstaste ausgelöst.

Der Trick besteht also darin, durch eine ungerade Anzahl von Gänsefuß-Eingaben diesen Modus herbeizuführen. Es geht ebenso durch Drücken der INSERT(INST)-Taste, allerdings nur für soviele Zeichen, wie oft sie gedrückt worden ist.

Im Gänsefuß-Modus erscheinen alle Steuer- und Funktionstasten in reverser Darstellung

Sie haben oben ein reverses Zeichen für die Funktionstasten erhalten. Die Zeichen für die Farben und Cursorbewegungen, also alle »gängigen« Funktionen, kennen Sie inzwischen sicher schon. Aber alle Steuer- und Funktionstasten?

Es gibt zwei Möglichkeiten, diese Zeichen zu finden:
Die 1. Methode verwendet entweder ganz primitiv im Direkt-Modus den Befehl: PRINT " mit nachfolgendem Drücken der Steuer- oder Funktionstaste oder sehr elegant den Dreizeiler
10 GET A$: IF A$=""THEN 10
20 PRINT CHR$(34) A$ CHR$(34) ASC(A$)
30 GOTO 10

Programm 3. ASCII-Code Wandler

```
3 REM****************
4 REM**CODE-WANDLER**
5 REM****************
10 PRINT CHR$(147)
20 INPUT "■ASCII-CODE";ACII
30 IF ACII=0 THEN END
40 IF ACII AND 128 THEN BILD=ACII AND 127 OR 64:GOTO 80
50 IF NOT ACII AND 64 THEN BILD=ACII:GOTO 80
60 IF ACII AND 32 THEN BILD=ACII AND 95:GOTO 80
70 BILD=ACII AND 63
80 PRINT TAB(5) "BILDSCH.CODE:"BILD
90 GOTO 20
```

ALLE TASTEN-, ZEICHEN- UND STEUER-CODES

Auch hier ist ein kleiner Pfiff drin. In Zeile 20 wird zuerst ein Gänsefuß (34) gedrückt, wodurch wir in dem danach benannten Modus sind. Das nachfolgende A$ erscheint im Fall einer Steuertaste als reverses Zeichen, nach dem abschließenden 2. Gänsefuß gibt uns die ASC-Funktion noch den ASCII-Code der gedrückten Taste.

Mit Gänsefüßen statt CHR$ hätten wir lediglich die beiden Zeichen A und $ auf den Bildschirm gedruckt. Mit dieser Methode erhalten Sie zum Beispiel für:

ASCII-Code 17 = [Q] Cursor Down ......

ASCII-Code 147 = [♥] CLR ....

Die 2. Methode ist viel einfacher. Schauen Sie in meine ASCII-Tabelle (Tabelle 1). Sie ist in Spalten zu je 32 Zeichen angeordnet. Die Steuerzeichen und die Farben stehen alle in Spalte 1 und 5.

Da finden Sie zum Beispiel über der Codezahl 17 die Funktion »Cursor-Down«. Wenn Sie jetzt in die 3. Spalte waagrecht rübergehen — also den Wert um 64 erhöhen — steht da das [Q]. Oder: In Spalte 5 ist der Taste CLR die Codezahl 147 zugeordnet. Zwei Spalten weiter (64 höher) steht das [♥].

Wenn Sie die Ergebnisse der 1. Methode oben mit den durch Spaltenhüpfen gefundenen Zeichen vergleichen, sehen Sie, daß es dieselben Zeichen sind, halt nur reversiert.

KOMBINATIONEN VC 20	KOMBINATIONEN C 64	1	2	3	4	5	6	7	8
		000	SPACE 032	@ 064	096	128	SHIFT SPACE 160	192	SHIFT SPACE 224
	CTRL — A	001	! 033	A a 065 065	A 097 097	BLK 129	161	A 193 193	225
	CTRL — B	002	" 034	B b 066 066	B 098 098	130	162	B 194 194	226
	CTRL — C	STOP 003	# 035	C c 067 067	C 099 099	LOAD RUN 131	163	C 195 195	227
	CTRL — D	004	$ 036	D d 068 068	D 100 100	132	164	D 196 196	228
	CTRL — E	WHT 005	% 037	E e 069 069	E 101 101	F1 133	165	E 197 197	229
CTRL — –	CTRL — F / CTRL — –	006	& 038	F f 070 070	F 102 102	F3 134	166	F 198 198	230
	CTRL — G	007	' 039	G g 071 071	G 103 103	F5 135	167	G 199 199	231
	CTRL — H	LOCK 008	(040	H h 072 072	H 104 104	F7 136	168	H 200 200	232
	CTRL — I	UN LOCK 009	) 041	I i 073 073	I 105 105	F2 137	169 169	I 201 201	233 233
	CTRL — J	010	* 042	J j 074 074	J 106 106	F4 138	170	J 202 202	234
	CTRL — K	011	+ 043	K k 075 075	K 107 107	F6 139	171	K 203 203	235
	CTRL — L	012	, 044	L l 076 076	L 108 108	F8 140	172	L 204 204	236
	CTRL — M	CAR RET 013	— 045	M m 077 077	M 109 109	SHFT RET 141	173	M 205 205	237
	CTRL — N	TEXT 014	. 046	N n 078 078	N 110 110	GRAF 142	174	N 206 206	238
	CTRL — O	015	/ 047	O o 079 079	O 111 111	143	175	O 207 207	239
	CTRL — P	016	0 048	P p 080 080	P 112 112	BLK 144	176	P 208 208	240
	CTRL — Q	CRSR DOWN 017	1 049	Q q 081 081	Q 113 113	CRSR UP 145	177	Q 209 209	241
CTRL — R	CTRL — R	RVS ON 018	2 050	R r 082 082	R 114 114	RVS OFF 146	178	R 210 210	242
	CTRL — S	CRSR HOME 019	3 051	S s 083 083	S 115 115	CLR SCRN 147	179	S 211 211	243
	CTRL — T	DEL 020	4 052	T t 084 084	T 116 116	INS 148	180	T 212 212	244
	CTRL — U	021	5 053	U u 085 085	U 117 117	BRN 149	181	U 213 213	245
	CTRL — V	022	6 054	V v 086 086	V 118 118	LT RED 150	182	V 214 214	246
	CTRL — W	023	7 055	W w 087 087	W 119 119	DARK GREY 151	183	W 215 215	247
	CTRL — X	024	8 056	X x 088 088	X 120 120	MED GREY 152	184	X 216 216	248
	CTRL — Y	025	9 057	Y y 089 089	Y 121 121	LT GRN 153	185	Y 217 217	249
	CTRL — Z	026	: 058	Z z 090 090	Z 122 122	LT BLU 154	186 186	Z 218 218	250 250
	CTRL — :	027	; 059	[091	123	LT GREY 155	187	219	251
	CTRL — £	RED 028	< 060	£ 092	124	PURP 156	188	220	252
	CTRL —]	CRSR RT 029	= 061	] 093	125	CRSR LEFT 157	189	221	253
	CTRL — ↑	GRN 030	> 062	↑ 094	126 126	YEL 158	190	222 222	254
	CTRL — =	BLU 031	? 063	← 095	127 127	CYAN 159	191	223 223	255 256

Tabelle 1. ASCII-Code

Tabelle 3. Bildschirm-Code

0–31	32–63	64–95	96–127	128–159	160–191	192–223	224–255
000 @	032 SPACE	064	096 SPACE	128 @	160	192	224
001 A a	033 !	065 ♠ A	097	129 A a	161 !	193 ♠ A	225
002 B b	034 "	066 B	098	130 B b	162 "	194 B	226
003 C c	035 #	067 C	099	131 C c	163 #	195 C	227
004 D d	036 $	068 D	100	132 D d	164 $	196 D	228
005 E e	037 %	069 E	101	133 E e	165 %	197 E	229
006 F f	038 &	070 F	102	134 F f	166 &	198 F	230
007 G g	039 '	071 G	103	135 G g	167 '	199 G	231
008 H h	040 (	072 H	104	136 H h	168 (	200 H	232
009 I i	041)	073 I	105	137 I i	169)	201 I	233
010 J j	042 *	074 J	106	138 J j	170 *	202 J	234
011 K k	043 +	075 K	107	139 K k	171 +	203 K	235
012 L l	044 ,	076 L	108	140 L l	172 ,	204 L	236
013 M m	045 -	077 M	109	141 M m	173 -	205 M	237
014 N n	046 .	078 N	110	142 N n	174 .	206 N	238
015 O o	047 /	079 O	111	143 O o	175 /	207 O	239
016 P p	048 0	080 P	112	144 P p	176 0	208 P	240
017 Q q	049 1	081 ● Q	113	145 Q q	177 1	209 ● Q	241
018 R r	050 2	082 R	114	146 R r	178 2	210 R	242
019 S s	051 3	083 ♥ S	115	147 S s	179 3	211 ♥ S	243
020 T t	052 4	084 T	116	148 T t	180 4	212 T	244
021 U u	053 5	085 U	117	149 U u	181 5	213 U	245
022 V v	054 6	086 V	118	150 V v	182 6	214 V	246
023 W w	055 7	087 ○ W	119	151 W w	183 7	215 ○ W	247
024 X x	056 8	088 ♣ X	120	152 X x	184 8	216 ♣ X	248
025 Y y	057 9	089 Y	121	153 Y y	185 9	217 Y	249
026 Z z	058 :	090 ♦ Z	122 ✓	154 Z z	186 :	218 ♦ Z	250
027 [	059 ;	091	123	155 [	187 ;	219	251
028 £	060 <	092	124	156 £	188 <	220	252
029]	061 =	093	125	157]	189 =	221	253
030 ↑	062 >	094 π	126	158 ↑	190 >	222 π	254
031 ←	063 ?	095	127	159 ←	191 ?	223	255

Machen wir die Probe:

Mit Methode 1 erhalten wir für »Rot« das reverse Pfund-Zeichen £. In der ASCII-Tabelle finden wir »Rot« unter 28. Zwei Spalten weiter, unter 28 + 64 = 92, steht dasselbe Zeichen.

Das gilt auch für alle CTRL-Kombinationen, nicht nur für die der Farben. Bei beiden Computern entspricht dem CTRL- ← das ←, beim C 64 erzeugt CTRL- : ein ;. Alle Kombinationen der Buchstaben mit CTRL erzeugen diese Buchstaben in reverser Darstellung.

Um das in einer kleinen praktischen Anwendung zu verdeutlichen, schlage ich vor, dieselbe Aufgabenstellung, die in Heft 5/84 sowohl mit Tastencode-Abfrage (Programm Nummer 1 auf Seite 104/105) als auch mit Tastaturpuffer-Abfrage (Programm Nummer 4 auf Seite 135) gelöst wurde, noch einmal zu verwenden, jetzt aber die Gänsefuß-Methode einzusetzen.

Um beim Eintippen des Programms unten sicherzustellen, daß alles klappt, habe ich statt der reversen Zeichen die Tasten angegeben beziehungsweise umrahmt, die nach dem 1. Gänsefuß gedrückt werden müssen.

Das Programm schaltet, wie die beiden anderen Versionen auch, die Bildschirm-Farben mit f-1, f-2, f-3 und @ um.

Programm 1. Abfrage mit Gänsefuß

```
410 PRINT "SHIFT und CLR/HOME
"
420 GET A$
430 IF A$= "" THEN 420
440 IF A$= " f-1 " THEN POKE
36879,126
450 IF A$= " f-2 " THEN POKE
36879,45
460 IF A$= " f-3 " THEN POKE
36879,25
470 IF A$= " @ " THEN POKE
36879,27
480 GOTO 420
```

Für den C 64 gelten in den Zeilen 440 bis 470 andere POKE-Adressen:

```
440 ..... POKE 53280,6:POKE 53281,7
450 ..... POKE 53280,5:POKE 53281,2
460 ..... POKE 53280,1:POKE 53281,1
470 ..... POKE 53280,3:POKE 53281,1
```

Ein letztes Problem bleibt uns noch. Wie schaffen wir es, daß wir im Gänsefuß-Modus auch Funktionen einsetzen können, die entweder keine eigene Taste haben (zum Beispiel 14 = Text, 8 = Lock) oder die beim Eintippen sofort die Funktion auslösen (zum Beispiel 13 = RETURN, 20 = DELete)?

ALLE TASTEN-, ZEICHEN- UND STEUERCODES

Hier müssen wir eine Methode anwenden, die meine Kinder und ich »finkeln« getauft haben und zwar deswegen, weil wir sie zum ersten und einzigen Mal vom Commodore-Software-Spezialisten Andy Finkel im amerikanischen Handbuch gefunden haben.

Sein Trick besteht darin, daß er in einer ASCII-Tabelle das entsprechende Zeichen für die Funktion heraussucht und es in mehreren Schritten an seinen vorgesehenen Platz bringt.

Ich will Ihnen zeigen, was ich damit meine:

Bitte, versuchen Sie mit der Gänsefuß-Methode die DELete-Taste in eine PRINT-Anweisung zu bringen
10 PRINT " INST/DEL "

Sie werden es nicht schaffen, da die DEL-Taste, statt ein reverses Zeichen zu drucken, ihrer Funktion nachgeht und das vorherige Zeichen löscht.

Jetzt »finkeln« wir:

1. Schritt:
10 PRINT " " (mit RETURN abschließen)

2. Schritt:
Mit dem Cursor auf die Leerstelle zwischen den Gänsefüßen fahren.

3. Schritt:
Aus der ASCII-Tabelle das Zeichen der DEL-Taste holen (T).

4. Schritt:
Die reverse Darstellung mit CTRL-REV.ON einschalten (der Cursor bleibt auf seiner Stelle) und das T drücken, mit RETURN abschließen. Jetzt steht das Zeichen drin und das Programm läuft.

Um Ihnen den Schritt 3 für alle widerborstigen Funktionen zu erleichtern, habe ich sie alle in der Tabelle 2 zusammengefaßt.

Da ich hoffe, daß Sie in Zukunft fleißig finkeln werden, muß ich Sie noch über einen lästigen Nebeneffekt aufklären, der bei ein paar Finkeleien auftritt. Einige der Funktionen, nämlich RETURN, DELete (schon wieder) und das SHIFT-RETURN wirken nicht nur im Programmablauf wie vorgesehen, sondern auch beim LISTen, was lästig sein kann. (Allerdings ergeben sich dadurch auch ungeahnte Möglichkeiten — siehe Artikel »Synthetische Steuerzeichen«. Das geSHIFTete RETURN (ASCII-Code 14) ist sehr nützlich bei Platz- und Speichermangel. Sie können nämlich mit " ■ " in einer langen Programmzeile den Cursor mit nur drei Zeichen auf den Anfang der nächsten Zeile bringen, mit CHR$(141) bräuchten Sie schon neun Zeichen, mit SPC(...) müssen Sie sehr genau die Cursorposition berechnen, mit einer entsprechenden Anzahl von »Cursor-Rechts«-Zeichen geht es auch nur mühsam. Also, nützlich ist SHIFT-RETURN durchaus!

Nur: Beim LISTen wird es auch ausgeführt und die Zeile, in der es steht, sieht recht blöd aus. Zusätzlich kann eine derart geLISTete Zeile nicht mehr geändert werden, sondern muß bei Verbesserungen völlig neu geschrieben und gefinkelt werden. Alles Gute hat seinen Preis!

Soviel sei zur Methode gesagt. Jetzt wollen wir zur Erholung und zur Übung ein kleines Spiel programmieren, in dem wir (fast) alles Gelernte auch anwenden.

Eine kleine Seltenheit ist bemerkenswert: Das Programm ist für VC 20 und C 64 identisch!

Die Spielaufgabe soll darin bestehen, die ersten sieben Buchstaben des Alphabets möglichst in der richtigen Reihenfolge auf den Bildschirm zu bringen.

BEDEUTUNG	ASCII-CODE	REVERSE DARSTELLUNG	FINKELN
LOCK (Sperre der Zeichensatz-Umschaltung	8	H	H
UNLOCK (Sperre aufheben)	9	I	I
RETURN	13	M	M
TEXT (2. Zeichensatz)	14	N	N
DEL (Zeichen löschen)	20	T	T
SHIFT RETURN (Cursor auf Anfang der nächsten Zeile)	141	■	SHIFT M
GRAF (1. Zeichensatz)	142	■	SHIFT N

Tabelle 2. Funktionen, die im Gänsefuß-Modus nur durch »Finkeln« eingetippt werden können

Das Finkel-System

Klingt einfach, aber die Buchstaben sollen in zufälliger Reihenfolge auftauchen. Zusätzlich hat der Spieler, falls der Buchstabe nicht der Reihenfolge entspricht, lediglich die Möglichkeit, ihn mit der DEL-Taste zurückzuweisen, wenn er schnell genug ist. Das Programm zählt die Felder und zeigt am Schluß das Ergebnis an.

Wir brauchen dazu:

— Einen Zufalls-Buchstaben-Erzeuger von A bis G (ASCII-Code 65 bis 71)
— einen Buchstaben-Drucker
— einen Buchstaben-Reihenfolgezähler
— eine Möglichkeit, die DEL-Taste zu drücken und damit den gedruckten Buchstaben rückgängig zu machen
— einen Fehlerzähler
— eine Prüfung, ob der letzte Buchstabe (71) erreicht ist.

Normalerweise müßte ich jetzt ein Flußdiagramm zeichnen und »strukturiert« vorgehen, so wie die ausgezeichnete Serie in diesem Heft lehrt. Man möge mir aber verzeihen, daß ich aus Erklärungsgründen in einzelnen Schritten vorgehe, welche uns erlauben, jederzeit Zwischenresultate mit Probeläufen zu überprüfen (Programm 2, siehe Listing).

Auf geht's!

TEIL 3

Den Buchstaben-Erzeuger und -drucker erhalten wir durch Zeile 50, welche für eine Variable A zufällige ASCII-Codes zwischen 65 und 71 erzeugt, sowie durch Zeile 70, die das Zeichen für den ASCII-Code ausdruckt.

```
50 A=INT(RND(0)*7)+65
70 PRINT CHR$(A);
```

Für weniger Versierte sei gesagt, daß RND(0) eine Zufallszahl zwischen 0 und 0,99 erzeugt, mit 7 multipliziert gibt das eine Zahl zwischen 0 bis 6,93. Die Funktion INT macht daraus eine ganze Zahl, zwischen 0 und 6, mit 65 addiert letztlich eine Zahl zwischen 65 und 71 — ASCII-Werte der Buchstaben A bis G.

Den Ausdruck der Buchstaben nebeneinander erreichen wir durch das Semikolon in Zeiel 70, die laufende Wiederholung durch einen Rücksprung in Zeile 320.

```
320 GOTO 50
```

Damit es nicht zu schnell geht, verzögern wir das Ganze mit einer Warteschleife in Zeile 80.

```
80 FOR T=1 TO 1000:NEXT T
```

Probieren Sie es mit RUN aus. Zeile 80 übrigens erlaubt Ihnen später den Schwierigkeitsgrad zu verändern.

Die geforderte richtige Reihenfolge der Buchstaben A, ich nenne sie hier R, setzen wir am Anfang auf 65 und erhöhen sie schrittweise um 1.

```
30 R = 65
310 R = R+1
```

In Zeile 60 prüfen wir, ob die Endzahl 71 für das G überschritten ist. Wenn ja, springen wir auf Zeile 400, mit der wir das Spielende anzeigen.

```
60 IF R > 71 THEN 400
400 PRINT "
SPIELENDE"
```

Bitte RUNnen Sie das Fragment wieder zur Probe.

Jetzt kommt die Beeinflussung der Reihenfolge mit der DEL-Taste. Wie gelernt fragen wir diese Taste mit einer GET-Schleife ab (Zeilen 100,110), ihre Lösch-Wirkung erreichen wir in Zeile 120 durch einen PRINT-Befehl (mit Semikolon!). Nach Drücken der DEL-Taste darf der Reihenfolge-Zähler der Zeile 310 natürlich nicht wirken, deshalb springen wir schon vorher aus der Zeile 210 zurück.

```
100 GET A$
110 IF A$<>"" THEN 300
120 PRINT "";
210 GOTO 50
```

Sie sehen oben, daß ich für die Abfrage der DEL-Taste die Finkel-Methode vorschlage. Die anderen Methoden gehen natürlich auch.

Nach RUN springt das Programm auf die noch nicht existierende Zeile 300 (was prompt zur Fehlermeldung führt), es sei denn, Sie drücken rechtzeitig die DEL-Taste.

In der Zeile 300 wollen wir prüfen, ob ein Fehler gemacht wurde, das heißt ob A mit der Reihenfolge R übereinstimmt. Im Fehlerfall wird die Fehlerzahl F um 1 erhöht. Vorher aber muß F auf 0 gesetzt werden.

```
20 F=0
300 IF A<>R THEN F=F+1
```

Sie können jetzt schon das Spiel üben. Aber es fehlen noch ein paar Feinheiten.

```
10 PRINT CHR$(147)
410 PRINT F "Fehler"
```

Zeile 10 ist klar, Zeile 410 druckt am Spielende die Fehlerzahl F aus.

Aber es gibt noch einen Fehler des Spielers, nämlich wenn er aus Versehen einen richtigen Buchstaben zurückweist. Deshalb fragen wir nach erfolgtem Drücken der DEL-Taste in den Zeilen 100 bis 120 nach, ob der Buchstabe tatsächlich falsch war. Wenn nicht, wird die Fehlerzahl F um 1 erhöht.

```
200 IF A=R THEN F=F+1
```

Damit uns nach RUN der erste Buchstabe nicht überrascht, verzögern wir sein Erscheinen mit

```
40 FOR T=1 TO 600: NEXT T
```

Zum Finkeln-Üben arrangieren wir die Anzeige des Spielendes und der Fehler etwas um. Alle Anweisungen sollen in nur einer Zeile stehen. Löschen Sie bitte die Zeile 410. In Zeile 400 wird gefinkelt und zwar mit dem Zeichen für SHIFT RETURN, welches laut Tabelle 2 mit SHIFT M erzeugt wird.

```
400 PRINT "        DAS
SPIEL IST ZU ENDE";"     "
F "FEHLER".
```

Bei LIST und bei Ausdruck mit einem Drucker sehen die gefinkelten Zeilen 110, 120 und 400 natürlich kurios aus und wie gesagt, sie lassen sich bei einem Tippfehler nicht korrigieren, sondern müssen neu geschrieben werden.

Der Bildschirm-Code

Der Vollständigkeit halber will ich noch die letzte der vorher genannten drei Methoden, ein Zeichen auf den Bildschirm zu bringen, erwähnen, insbesondere, weil der dabei verwendete Bildschirm-Code (auch Video-Code genannt) oft zu Verwechslungen mit dem ASCII-Code führt.

Auf Anhieb ist es auch nicht einzusehen, warum Commodore einen anderen Code verwendet, wenn ein Zeichen direkt auf den Bildschirm — oder genauer gesagt in den Bildschirm-Speicher — gePOKEt werden soll.

Der Grund dafür liegt darin, daß dem ASCII-Code nicht nur Zeichen zugeordnet sind, sondern auch Farben und Funktionen. Außerdem sind im ASCII-Code die reversen Zeichen nicht enthalten, sondern müssen — wie Sie ja inzwischen wissen — jeweils umgeschaltet werden. Das alles ist für ein Betriebssystem viel zu kompliziert.

Es ist viel einfacher, im Festspeicher (ROM) alle Zeichen der zwei Zeichensätze fest zu verankern, von wo sie das Betriebssystem herausholen und auf den Bildschirm bringen kann.

Die Reihenfolge der Zeichen und ihr Code sehen Sie in der Tabelle 3. Sie ähnelt in mehreren Bereichen der ASCII-Reihenfolge, einige Spalten sind sogar identisch. Das macht eine Umrechnung — auch für das Betriebssystem — sehr einfach.

Folgende Blöcke der beiden Codearten entsprechen einander:

ASCII-CODE	BILDSCHIRM-CODE
0 - 31	entspricht keinem Zeichen
32 - 64	32 - 64
64 - 95	0 - 31
96 - 127 192 - 223	64 - 95
128 - 159	entspricht keinem Zeichen
160 - 191 224 - 255	96 - 127
entspricht keinem ASCII-Code	128 - 255

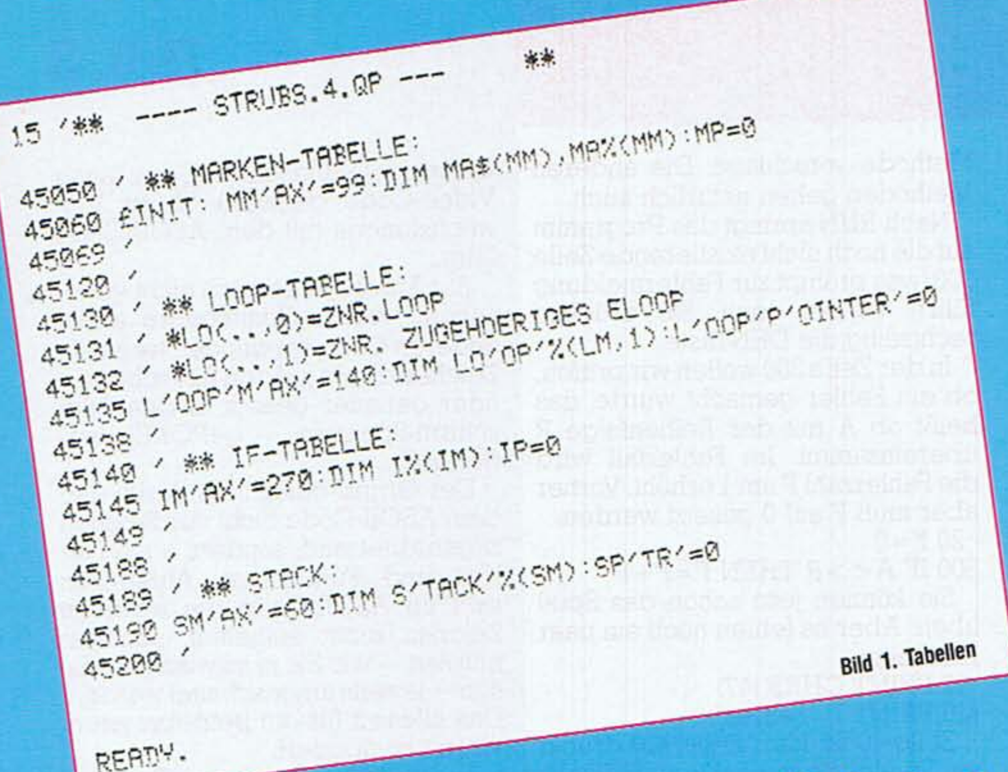
Ein Programm zur Umrechnung von ASCII-Code in Bildschirm-Code sieht dementsprechend aus wie in Programm 3 (VC 20 und C 64).

Dabei habe ich als Variable gewählt:

— ACII = ASCII-Code
— Bild = Bildschirm-Code

Im Hinblick darauf, daß unser Hauptthema die Abfrage der Tastatur ist, soll uns dieser Ausflug genügen.

(Dr. Helmut Hauck)



```
15 '**  ---- STRUBS.4.QP ---         **
45050 ' ** MARKEN-TABELLE:
45060 £INIT: MM'AX'=99:DIM MA$(MM),MA%(MM):MP=0
45069 '
45120 '
45130 ' ** LOOP-TABELLE:
45131 ' *LO(..,0)=ZNR.LOOP
45132 ' *LO(..,1)=ZNR. ZUGEHOERIGES ELOOP
45135 L'OOP'M'AX'=140:DIM LO'OP'%(LM,1):L'OOP'P'OINTER'=0
45138 '
45140 ' ** IF-TABELLE:
45145 IM'AX'=270:DIM I%(IM):IP=0
45149 '
45188 '
45189 ' ** STACK:
45190 SM'AX'=60:DIM S'TACK'%(SM):SP'TR'=0
45200 '

READY.
```

Bild 1. Tabellen



Ein Precompiler für Basic-Programme (Teil 4)

In der heutigen letzten Folge wollen wir einige Teile des Programmes Strubs genauer ansehen und untersuchen, wie man das Programm um zusätzliche Funktionen erweitern kann. Dabei werden wir sehen, daß ein solches Übersetzungsprogramm auch für ganz andere Aufgaben eingesetzt werden kann.

In den vorausgehenden Folgen wurde bereits erwähnt, daß die strukturierte Programmierung vor allem Vorteile in bezug auf Wartung, Änderungen und Erweiterbarkeit von Programmen bietet. Dies gilt auch für das Programm Strubs. Um in den Genuß dieser Vorteile zu gelangen, ist allerdings der Zugang zum Quellprogramm erforderlich. Wenn Sie sich das in Heft 5 abgedruckte Objektprogramm ansehen, werden Sie feststellen, daß es auch nicht viel aussagekräftiger als ein unkommentiertes Assemblerlisting ist. Wenn Sie an der Entwicklung eigener Programmerweiterungen interessiert sind, sollten Sie sich deshalb beim Verlag das Quellprogramm besorgen. Da ich hier davon ausgehen muß, daß die meisten Leser das Quellprogramm nicht besitzen, lohnt es sich gar nicht erst, systematisch die einzelnen Programmteile vorzustellen.

Statt dessen wollen wir nur die für Programmerweiterungen wichtigsten Programmelemente vorstellen und anhand einiger exemplarischer Erweiterungen, die auch, ohne sich weitere Gedanken zu machen, einfach eingetippt werden können, aufzeigen, wie man Erweiterungen implementieren kann und was dabei zu beachten ist. Aus dem gleichen Grund geben wir nur die Änderungen an, die im Objektprogramm vorzunehmen sind. Eine Anpassung an das Quellprogramm dürfte keine Probleme bereiten.

Achten Sie bei allen Programmänderungen darauf, daß das geänderte Programm abgespeichert wird, bevor es zum ersten Mal gestartet wird, da das Programm den Zeiger auf das Programmende verstellt. Sollte das Programm durch Erweiterungen so lang werden, daß es in den Editbereich hineinreicht, kann der Anfang des Editbereichs in Schritten zu 256 Byte nach oben verschoben werden, um Platz zu schaffen. Dazu ist in den Zeilen 70 bis 80 die Zahl 40 überall, wo sie auftaucht, durch eine größere Zahl (jeweils 4 für jedes Kilobyte) zu ersetzen (vgl. auch den Schluß der 3. Folge).

Die wichtigsten Programmelemente

Eine grobe Übersicht über den Aufbau des Programms haben wir bereits in der 2. Folge gegeben. Bevor wir uns nun mit einzelnen Erweiterungen beschäftigen, wollen wir zunächst einmal die wichtigsten Programmelemente vorstellen, die man für Änderungen und Erweiterungen des Programms benötigt. Wie bereits erwähnt, liest Strubs das Quellprogramm zweimal vom Anfang bis zum Ende durch. Um Zeit zu sparen, wird im 1. Lauf nur jeweils der Anfang einer Zeile untersucht. Deshalb müssen alle Befehle, die bereits im 1. Lauf zu behandeln sind, auch am Anfang einer Zeile stehen, während Befehle, die nur im 2. Lauf behandelt werden, überall stehen können. Ein Beispiel:

Die Definition von Marken muß am Zeilenanfang erfolgen, während der Aufruf von Marken an jeder Stel-



le erfolgen kann. Die Aufgabe des 1. Laufs besteht darin, verschiedene Tabellen anzulegen, mit deren Hilfe dann im 2. Lauf das endgültige Objektprogramm erzeugt wird.

Jede dieser Tabellen besteht aus einem oder mehreren Array(s), einer Variablen, deren zweiter Buchstabe ein »M« für »Maximal« ist und die Dimension, das heißt die maximale Zahl von Einträgen festlegt, und aus einer Variablen, deren zweiter Buchstabe ein »P« für »Pointer« ist und die auf den jeweils nächsten freien Listenplatz zeigt. Bei Speicherplatzproblemen brauchen nur die Werte der Dimensionsvariablen im Init-Teil geändert zu werden. Möchte man zum Beispiel mehr als 99 Marken (die jetzige Maximalzahl) benutzen, dann schreibt man in Zeile 45060 zum Beispiel »MM = 150:...«.

Die Tabellen werden in den Zeilen 45050 bis 45200 definiert (Bild 1). Die Dimension des Stacks bestimmt die mögliche Schachtelungstiefe. Dazu kommen die Tabellen der neuen Befehle (Zeile 45260 bis 45274) und der Fehlermeldungen (Zeile 45480 bis 45514).

Dem schrittweisen Lesen des Quellprogramms dienen die Variablen C und NC. Die Variable C enthält den Code des jeweils zuletzt gelesenen Zeichens, wobei der Wert 0 ein Zeilenende markiert. Die Variable NC enthält die Adresse des nächsten zu lesenden Zeichens. Im 2. Lauf wird zeilenweise das Objektprogramm erzeugt, wobei die jeweils aktuelle Zeile in der Variablen Z$ aufgebaut wird. Dabei enthalten die beiden ersten Zeichen von Z$ Low- und Highbyte der Zeilennummer (so wie sie später im Speicher steht), und das letzte Zeichen der fertigen Zeile besteht aus dem Zeichen CHR$(0).

Die relevanten Zeichencodes, auf die Strubs reagiert, werden in den Zeilen 45240 bis 45254 definiert (Bild 2). Die Variable ZA enthält die Adresse des Anfangs der Zeile, die gerade bearbeitet wird. In EA steht die Startadresse des Editbereichs.

Damit kommen wir zu den für Erweiterungen wichtigen Modulen von Strubs. Die Prozedur »NEXTCHAR« sucht ab Adresse NC das nächste relevante Zeichen des Quellprogrammtextes und liefert dessen Code in der Variablen C. Dabei werden Leerzeichen (Zeile 250) und Kommentare (Zeile 280-295) überlesen. Strings werden direkt in die Ausgabezeile Z$ übertragen (Zeile 350). Der Zeiger NC wird auf das nächste zu lesende Zeichen gesetzt. Die Prozedur »HOLNAME« (Zeile 750-830) liest ab aktueller Adresse NC einen Namen (zum Beispiel Befehl, Label) und zwar bis eines der Trennzeichen »:«, »,«, Blank oder Zeilenende erscheint. Der Name wird in der Variablen T$ ausgegeben, C enthält das erste relevante Zeichen hinter dem Namen (das ist außer beim Blank das Trennzeichen), und NC zeigt auf das nächste Zeichen.

Die Prozedur »SCHREIBZEILE« (Zeile 550-580) generiert auf der Diskette aus den nacheinander eingegebenen Zeilen Z$ das zusammenhängende Objektprogramm und gibt die Nummer der aktuellen Zeile auf dem Bildschirm aus. Die Variable AA (Linkadresse) darf außerhalb dieser Routine nicht verändert werden!

Die Prozedur »ERROR« (Zeile 8050 bis 8099) erwartet als Eingabe einen Fehlercode ER. Dabei handelt es sich um den Index der Fehlermeldung in der Tabelle der Fehlermeldungen. Die Zeilennummer und die Fehlermeldung werden auf dem Bildschirm ausgegeben und zugleich in eine Fehlertabelle eingetragen, die man sich nach der Übersetzung auf Bildschirm oder Drucker ausgeben lassen kann. Zusätzlich wird die Fehlermeldung in die Ausgabezeile Z$ geschrieben, so daß sie auch im Objektprogramm erscheint. Die Übersetzung wird mit der folgenden Zeile fortgesetzt.

Die Prozedur »ABBRUCH« (Zeile 50000 bis 50030) sorgt für einen kontrollierten Abbruch der Übersetzung. Sie erwartet ebenfalls als Eingabe den Fehlercode ER und gibt die entsprechende Fehlermeldung aus. Danach wird die Tabelle der bisher bemerkten Fehler ausgegeben, offene Files ordnungsgemäß geschlossen und Strubs neu gestartet.

```
45240 ' ** RELEVANTE ZEICHENCODES **
45250 DP=ASC(":"):KO'MMENTAR'=ASC("'"):LA'BEL'=ASC("£"):NU$=CHR$(0):BL=ASC(" ")
45253 BE'FEHL'=ASC("!"):TE'XT("")'=34:G'O'T'O-CODE'$=CHR$(137)
45254 I'F'C'ODE'$=CHR$(139):TH'EN-CODE'=167:NO'T'$=CHR$(168):K'OM'M'A-CODE'=44
```

Bild 2. Relevante Zeichencodes

```
10 REM ****  LISTER-DEMO **************
30 PRINT"[illegible]TEST[illegible]"
20 REM WIRD ZU:
30 PRINT"<CD><CR><CU><CL><RON><ROF><HO>TEST<DEL><INS><WHT><RED><GRN><BLU><BLK><P
UR><YEL><CYN>"

READY.
```

Bild 3. Beispiellister

Die Prozedur »WARTEN« (Zeile 49550 bis 49570) fordert den Benutzer auf, eine Taste zu drücken und wartet auf den Tastendruck.

Die Prozedur »INIT« (Zeile 45050 bis 45999) enthält die Definition der Variablen und Tabellen sowie die Interpretererweiterung.

Im »MENÜ« (Zeile 40050 bis 40495) können die verschiedenen Funktionen angewählt werden.

Die Prozeduren »BEFEHLE IM 1. LAUF« (Zeile 1550-2497) und »BEFEHLE IM 2. LAUF« (Zeile 2550-3640) werden von Strubs aufgerufen, sobald im Quellprogramm das Erkennungszeichen »!« für Befehle (Code in der Variablen BE) entdeckt wird. Sie holen den Namen des Befehls, suchen diesen in der Befehlstabelle und rufen entsprechend dem Index (+1) des Befehls in dieser Tabelle ein Unterprogramm auf. Falls der Befehl nicht in der Tabelle gefunden wird, wird eine entsprechende Fehlermeldung ausgegeben. Im 1. Lauf kommt noch die Ausgabe der Blockstruktur hinzu. Hierzu dient die Variable In (für Indentmodus). IN = 0 bedeutet, auf der gleichen Schachtelungsebene zu bleiben.

Damit haben wir nun das notwendige Wissen zusammen, um an dem Programm Strubs einige Änderungen und Erweiterungen vorzunehmen.

Andere Anwendungen

Bei den Programmtexten, die Strubs übersetzt, handelt es sich zwar um erweiterte Basicprogramme, aber nichtsdestoweniger um Basicprogramme. Deshalb ist es relativ einfach, Strubs auch zur Bearbeitung ganz normaler Basic-Programme einzusetzen. Zwei sinnvolle Möglichkeiten wollen wir im folgenden vorstellen.

1. Ein SPEED-UP-Programm, um normale Basicprogramme schneller zu machen.
2. Ein Programm, das besser lesbare Listings erstellt.

Dabei ist zu beachten, daß die Änderungen, die wir dazu vornehmen, nicht wie die Makro-Funktion eine Erweiterung des eigentlichen Programmes Strubs und seiner Funktion darstellen, sondern daß wir zwei völlig neue Programme mit völlig neuen Aufgaben erhalten. Deshalb sollten auch die erhaltenen Programme unter neuen Namen, beispielsweise »SPEED-UP« und »LISTER«, abgespeichert werden. Das Arbeiten mit diesen Programmen unterscheidet sich nicht von der Arbeit mit dem »normalen« Strubs-Programm.

Schnelleres Basic

Zunächst wollen wir Strubs so ändern, daß es normale Basicprogramme in Programme übersetzt, die keine Leerzeichen und Kommentare mehr enthalten und dadurch schneller ablaufen. Wie Sie sich erinnern werden, benutzt Strubs für Kommentare, die gelöscht werden sollen, ein eigenes Zeichen »'«. Kommentare, die mit REM gekennzeichnet werden, bleiben im Objektprogramm erhalten. Da Strubs bereits alle Blanks entfernt (außer in Strings), brauchen wir nur noch dafür zu sorgen, daß Strubs auf das REM-Token reagiert wie bisher auf das Kommentarzeichen »'«. Die relevanten Zeichencodes, auf die Strubs reagiert, werden in den Zeilen 45250 bis 45254 definiert (Bild 2). Wir brauchen nur in Zeile 45250 das KO = ASC(»'«) durch KO = 143 (143 ist das REM-Token) ersetzen und schon ist das Speed-Up-Programm fertig. Genauso können Sie die Erkennungszeichen für Label und die neuen Befehle ändern. Dies ist, um Kon-

```
10 REM ***** MAKROS BEISPIELE ********
20 !DMAKRO:STOPAN POKE 788,49
30 !DMAKRO:STOPAUS POKE 788,52
40 !DMAKRO:READJOY JS=PEEK(56320)
50 !DMAKRO:WARTE POKE 198,0:WAIT 198,1
60 !DMAKRO:CLOSEALL SYS 65511
70 !DMAKRO:SPRITEXPOS POKE 53248+2*
80 !DMAKRO:SPRITEYPOS POKE 53249+2*
90 !DMAKRO:SPRITEAN POKE 53269,PEEK(53269) OR 2↑
95 '                    ************
96 '   ********   AUFRUFE:
100 !M.STOPAUS:!M.CLOSEALL:!M.STOPAN
110 !M.READJOY:PRINT JS
120 !M.SPRITEAN 5:!M.WARTE
130 !M.SPRITEXPOS 5,240:!M.SPRITEYPOS 5,170
140 !M.FEHLER

READY.

10 REM****MAKROSBEISPIELE********
100 POKE788,52:SYS65511:POKE788,49
110 JS=PEEK(56320):PRINTJS
120 POKE53269,PEEK(53269)OR2↑5:POKE198,0:WAIT198,1
130 POKE53248+2*5,240:POKE53249+2*5,170
140 ***** ERR:UNDEFINIERTES MAKRO********

READY.
```

Bild 4. Beispiele Makros

```
0 ********* SPEED-UP ****************
45250 DP=ASC(":"):KO=143:LA=ASC("£"):NU$=CHR$(0):BL=ASC(" ")

READY.

0 ***** LISTER   ******************
350 Z$=Z$+C$(C):NC=NC+1:C=PEEK(NC):IF C AND C<>TE THEN 350
45300 DIM C$(255):FOR I=0 TO 255:C$(I)=CHR$(I):NEXT
45310 C$(17)="<CD>":C$(29)="<CR>":C$(145)="<CU>":C$(157)="<CL>"
45312 C$(18)="<RON>":C$(146)="<ROF>":C$(19)="<HO>":C$(147)="<CLR>"
45314 C$(20)="<DEL>":C$(148)="<INS>"
45316 C$(5)="<WHT>":C$(28)="<RED>":C$(30)="<GRN>":C$(31)="<BLU>"
45318 C$(144)="<BLK>":C$(156)="<PUR>":C$(158)="<YEL>":C$(159)="<CYN>"

READY.

0 ********* MAKROS ****************
1571 IF I>14 THEN ON I-14 GOSUB 2350,2380
2350 IF NP>NM THEN ER=10:GOTO 50000
2355 Z$="":GOSUB 750:NA$(NP,0)=T$
2360 Z$=Z$+CHR$(C):GOSUB 250:IF C<>0 THEN 2360
2370 NA$(NP,1)=Z$:NP=NP+1:IN=0
2375 RETURN
2380 IN=0:RETURN
2571 IF I>14 THEN ON I-14 GOSUB 3700,3750
3700 Z$="":C=0:RETURN
3750 GOSUB 750
3755 FOR I=0 TO NP:IF NA$(I,0)<>T$ THEN NEXT
3760 IF I>=NP THEN ER=11: GOTO 8050
3765 Z$=Z$+NA$(I,1):RETURN
45155 NM=40:DIM NA$(NM,1):NP=0
45265 BM=15:DIM BE$(BM)
45275 DATA DMAKRO,M
45480 EM=11:DIM ER%(30,1):EP=0:DIM ER$(EM)
45500 FOR I=0 TO EM:READ ER$(I):NEXT
45515 DATA "ZU VIELE MAKROS","UNDEFINIERTES MAKRO"
```

Bild 5. Die besprochenen Erweiterungen auf einen Blick

flikte zu vermeiden, für den Fall sinnvoll, daß Sie mit Strubs Programme für Interpretererweiterungen übersetzen, die ihrerseits »!« oder das Pfundzeichen als Erkennungszeichen für ihre neuen Befehle benutzen.

Listings

Wollen Sie im »64'er« eigene Programme veröffentlichen? Dann können Sie den Lesern viel Ärger ersparen, wenn Sie das Listing vorher mit dem Programm »LISTER« aufbereiten. »LISTER« übersetzt Basic-Programme in Programmtexte, in denen die schwer entzifferbaren Steuer- und Grafikzeichen innerhalb von Strings durch lesbare Worte »⟨CDOWN⟩« oder »⟨HOME⟩« ersetzt sind (Bild 3).

Dazu ändern wir eine Zeile innerhalb der Prozedur »NEXTCHAR«. In Zeile 350 werden gelesene Zeichen mit dem ASCII-Code C innerhalb von Strings direkt in die Ausgabezeile Z$ übertragen. Wenn wir nun in Zeile 350 Z$=Z$+CHR$(C) durch Z$=Z$+C$(C) ersetzen, dann können wir ein Array C$(255) definieren, das in jedem ASCII-Wert den String enthält, der dafür im Objektprogramm erscheinen soll. Die Definition dieses Arrays gehört in das Modul »INITIALISIERUNG«:

45300 DIM C$(255):FOR I=0 TO 255:C$(I)=CHR$(I):NEXT

Damit haben wir zugleich unser Array mit den normalen Werten vorbesetzt. Jetzt bleiben nur noch die Ersetzungen:

45310
C$(17)="⟨CDOWN⟩":C$(19)="⟨HOME⟩"
45312
C$(28)="⟨ROT⟩":C$(31)="⟨BLAU⟩"
... usw.

Hier können Sie nun jedem Zeichen ein beliebiges Wort zuordnen: Den ASCII-Code der einzelnen Zeichen finden Sie im C-64 Handbuch auf S. 135 oder Sie können ihn einfach durch Eingabe von
PRINT ASC("X")
feststellen, wobei »X« für das interessierende Zeichen steht. Bei sehr vielen Zeichen innerhalb eines Strings kann es allerdings vorkommen, daß die Zeilen zu lang werden. Deshalb sollten die Worte möglichst kurz gewählt werden.

Makros

An einem etwas umfangreicheren Beispiel wollen wir nun zeigen, wie man neue Strubs-Befehle implementiert und wie man die Prozeduren von Strubs benutzen kann. Dies soll am Beispiel einer Makro-Funktion demonstriert werden.

Makros, vor allem von Assemblern her bekannt, stellen so etwas wie Abkürzungen für kurze Programmausschnitte dar. Dadurch verringert sich die Tipparbeit und vor allem werden die Quellprogramme übersichtlicher.

In der Makro-Definition wird ein Makro-Name definiert und diesem ein Programmstück zugeordnet. Überall, wo nun im Quellprogramm ein Makro aufgerufen wird, erscheint im Objektprogramm an dieser Stelle das entsprechende Programmstück. Ein einmal definiertes Makro kann wie ein Label beliebig oft aufgerufen werden.

Für die Definition eines Makros wollen wir den Befehl »!DMAKRO« und für den Aufruf eines Makros den Befehl »!M« wählen. Ein Beispiel mag die Wirkungsweise der neuen Befehle demonstrieren:

10 !DMAKRO:NAME SYS 833:X=PEEK (878)
...
200 PRINT X:!M,NAME:PRINT X

Die Definitionszeile 10 wird gelöscht, da sie nur für die Übersetzung notwendige Informationen enthält. Die Zeile 200 mit dem Makro-Aufruf sieht im Objektprogramm folgendermaßen aus:

200 PRINTX:SYS833:X=PEEK(878):PRINTX

Einige Beispiele für Makros und deren korrekte Benutzung sowie das sich ergebende Objektprogramm zeigt Bild 4. Vor allem ist darauf zu achten, daß Makronamen wie alle Befehls- und Labelnamen mit einem der Trennzeichen abgeschlossen werden müssen. Insbesondere darf bei der Makrodefinition und beim Aufruf mit nachfolgenden Parametern (Spritemakros in Zeile 120 und 130) nicht das Blank hinter dem Makronamen vergessen werden! Jede Makrodefinition benötigt eine eigene Zeile. Eine Übergabe von

Parametern an ein Makro ist nicht möglich. Achten Sie bei der Arbeit mit Makros darauf, daß die entstehenden Zeilen des Objektprogramms nicht zu lang werden. Zeilen, die länger als 80 Zeichen sind, lassen sich nicht mehr editieren. Zeilen, die länger als 256 Zeichen werden, führen zum unkontrollierten Abbruch der Übersetzung mit »String too long error«. In diesem Fall kann man mit »GOTO 50000« die Nummer der verantwortlichen Zeile erfahren und offene Files schließen.

Um die Übersetzung zu ermöglichen, muß im 1. Lauf eine Tabelle der Makronamen und der zugehörigen Programmausschnitte angelegt werden. Im 2. Lauf werden dann alle Aufrufe durch den zugehörigen Text ersetzt. Die Verteilung auf zwei Läufe bietet den Vorteil, daß ein Makro (ebenso wie Labels) auch schon vor der Definition aufgerufen werden kann.

Zur Implementation sind folgende Schritte erforderlich: Zunächst muß dem Übersetzungsprogramm mitgeteilt werden, daß es zwei neue Befehle gibt. Dann müssen wir die notwendige Tabelle definieren und auch entsprechende Fehlermeldungen vorsehen. Diese Erweiterungen gehören in den INIT-Teil.

Schließlich muß noch dafür gesorgt werden, daß Strubs weiß, wie es im 1. und 2. Lauf auf die neuen Befehle zu reagieren hat.

Die Befehlstabelle wird in Zeile 45265 definiert. Hier erhöhen wir die Zahl der Befehle um 2 und fügen dann noch eine DATA-Zeile mit den beiden neuen Befehlsnamen ein:

```
45265 BM=15:...
45275 DATA DMAKRO,M
```

Wählt man Befehlsnamen, die reservierte Basic-Worte enthalten, dann müssen die Tokens berücksichtigt werden (wie dies für IF in der Zeile 45271 geschieht). Für einen Befehl »DEFMAKRO« wäre zum Beispiel

```
BE$(14)=CHR$(150)+"MAKRO"
```

zu setzen (150 = DEF-Token).

Für die Tabelle wählen wir ein Array NA$(NM,1), da der Name M bereits für die Markentabelle vergeben ist. Die Dimension (..,0) soll die Namen und die Dimension (...,1) den zugehörigen Text aufnehmen.

```
45155 NM=40:DIM NA$(NM,1):NP=0
```

```
--------------------------------------------------------------------------------
CROSS REFERENCE MAP                          STRUBS.4.0P                    PAGE
--------------------------------------------------------------------------------
AA          560*   565    570   5120*  5130
AD()        555   1125   1574   1605   2100   2260   5052   5920   6100   6550
6655   8050   8060  45410  50008
B$         1565*  1575   1577   1579   1581   2685*  2693  48060* 48070  49060*
49070  49560*
BE         3470  45253*
BE$()      1550  45641  45265  45270* 45271*
BL          250    295    795    820  45250*
BM         1565  45641   2565  45265* 45270
C           260*   265    280*   290*   295    350*   795*   800    820   2420
2425   2470   2590   3010   3036*  3400*  3470   3495*  3610   4100
4108   4115   4130   4360   4380   5580*  5585   8090*
C$         8090*
DI         1170   1605   2100   2260   2640   2693   3030   3090   3490   3630
6100   8060  45220* 48140  49140
DP          795   5585  45250*
E          5090*  5095
E$         5090*  5095
EA           80*  5052   5120   5555   6550   8950
EM         8060  45480*
EP         5180   8060* 45480* 49050  49110  49120
ER         1160*  1565*  1600*  1605*  1640*  2010*  2040*  2100*  2160*  2275*
2410*  2425*  2565*  5143*  6050*  8050   8060   8080  50008
ER$()      8050   8080  45480  45500* 49140  50008
ER%()      8060* 45480  49140
F$         5070*  5080
GT$        2640   2685   3090  45253*
H          1125*  1170*  1180   2260*  2300   2425*  2647*  2648   3100*  3130
3190*  3200  48055* 48102* 48150  49055* 49102* 49150
H%          565*   570
I          1140*  1160   1170   1550   1565   1569*  1570   2275*  2290   2300
45641   2565   2570  45270* 45500* 45600* 45610* 45650* 48120* 48140
48150  49120* 49140  49150
I%()       2100*  2300*  3030   3090  45145
IC$        3010   3600  45254* 45271
IM        45145*
IN         1575   1577   1579   1581   1615*  1660*  1680*  2025*  2052*  2107*
2240*  2320*  2485*  3620*
IP         2010*  2050*  3030   3036*  3090   3140*  6560* 45145*
KM          795  45254*
KO          265    280   6585  45250*
L          2647*  2648   3100*  3130   3190*  3200
LA         4100   4360  45250*
LM         1605  45135*
LO%()      1605*  2640   2693   3490   3630  45135
LP         1605*  2595*  6560* 45135*
MA$()      1140   6100* 45060  48140
MA%()      1170   6100* 45060  48140
MM         2410   6050  45060*
MP         1140   1160   2410   2460*  6050   6100* 45060* 48050  48120
NC          250*   260    280*   290*   350*   370*   795    800*   820*  4080*
4115*  5580*
NO$        3010   3600  45254*
NU$        2640   3090   4115   8090  45250*
S%()       2050*  2100   2275   2595*  2640   2693   3490   3630  45190
SM         1600   2010   2160  45190*
SP         1600   1605*  1640*  2010*  2040   2050   2100   2160*  2275   2595*
2630*  2640   2693   3490   3620*  3630   5143   6560* 45190*
T$          750*   800*  1120   1140   1550  45641   6100
TA         1575   1577*  1579   1581*  5136*
TE          350  45253*
TH         3030   3480   3630  45254*
W         45600* 45610* 45650*
X         45410
Z$          350*   550    560    570   1180*  2590*  2640*  2648*  2685   2693*
3010*  3030*  3090*  3130*  3200*  3400*  3470*  3480*  3490*  3600*
3610*  3630*  4060*  4108*  4115*  4380*  5096*  5099   8080* 40160*
40170  40180  40195
Z1         6550*  6655*  6660
ZA          555   1125   1574   1605   2100   2260   2647   3100   3190   4060
4080   5555*  5570   5580   5920*  6100   6550*  6585   6655*  8050
8060  50008
```

Bild 6. Variablenliste

Damit können 41 Makros definiert werden. Indem wir die Zahl der Fehlermeldungen von 9 auf 11 erhöhen, erhalten wir die beiden neuen Fehlercodes 10 und 11 für »zu viele Makros« und »undefiniertes Makro«.

```
45480 EM=11:DIM...
45500 FOR I=0 TO EM:READ ...
45515 DATA "ZU VIELE MAKROS",
"UNDEFINIERTES MAKRO"
```

Nun müssen wir in die beiden Module »BEFEHLE IM 1. LAUF« beziehungsweise »BEFEHLE IM 2. LAUF« jeweils zwei Routinen für die neuen Befehle einfügen. Da die beiden Verteilerzeilen bereits voll sind, legen wir zwei neue Verteilerzeilen an, die dann aber auch gleich für 10 weitere neue Befehle Platz bieten:

```
1571 IF I>14 THEN ON I-14 GOSUB
2350,2380
```

für den 1. Lauf und

```
2571 IF I>14 THEN ON I-14 GOSUB
3700,3750
```

für den 2. Lauf.

Die Routine für »!DMAKRO« im 1. Lauf soll zunächst prüfen, ob noch Platz in unserer Makro-Tabelle ist und, falls nicht, mit entsprechendem Fehlercode die Abbruch-Routine anspringen:

```
2350 IF NP>NM THEN ER=10:
GOTO 50000
```

Jetzt können wir mit Hilfe der Prozedur »HOLNAME« den Makro-Namen lesen und in unserer Tabelle speichern:

```
2355 Z$="":GOSUB750:NA$(NP,0)
=T$
```

Nun übertragen wir den Rest der Definitionszeile mit Hilfe von »NEXTCHAR« nach Z$ (dadurch werden auch Strings mit übertragen. Als Ausgabezeile dient Z$ ja erst im 2. Lauf).

```
2360 Z$=Z$+CHR$(C):GOSUB
250:IF C<>0 THEN 2360
```

Nun brauchen wir nur noch den Text in die Tabelle aufzunehmen, den Zeiger zu erhöhen und den Indentmodus angeben.

```
2370 NA$(NP,1)=Z$:NP=NP+1:IN=
0
2375 RETURN
```

Der Aufruf eines Makros interessiert im 1. Lauf nicht, also:

```
2380 IN=0:RETURN
```

Im 2. Lauf soll die Definitionszeile gelöscht werden. Dazu löschen wir den Ausgabestring und weisen C den Code für Zeilenende zu:

```
3700 Z$="":C=0:RETURN
```

Beim Aufruf eines Makros mit »!M« holen wir zunächst den Namen des Makros mit »HOLNAME« und suchen ihn in der Tabelle:

```
3750 GOSUB 750
3755 FOR I=0 TO NP: IF NA$(I,0)
<>T$ THEN NEXT
```

Falls der Name nicht gefunden wird, erfolgt ein Sprung zur Error-Routine mit dem Code für »undefiniertes Makro«:

```
3760 IF I>NP THEN ER=11: GOTO
8050
```

Nun ist nur noch das definierte Programmstück in die Ausgabezeile zu übertragen:

```
3760 Z$=Z$+NA$(I,1):RETURN
```

Dadurch, daß diese Makro-Erweiterung Zeile für Zeile besprochen wurde, um zu zeigen, wie man die von Strubs vorgegebenen Prozeduren benutzen kann, ist vielleicht der Eindruck entstanden, eine solche Erweiterung sei relativ kompliziert. Wenn Sie sich aber das Ganze noch einmal genauer ansehen, können Sie feststellen, daß für die Implementation neuer Befehle im Prinzip nur drei Schritte erforderlich sind:

1. Eintrag der neuen Befehlsnamen in die Befehlstabelle
2. Einfügen der entsprechenden Routinen
3. Eintrag der Adressen dieser Routinen in die beiden Verteilerzeilen

Die ganze Arbeit des Suchens und Decodierens übernimmt Strubs automatisch.

Wie neue Funktionen (beispielsweise die Ausgabe der Makro-Tabelle) in das Menü aufgenommen werden können, haben sie bereits in der letzten Folge am Beispiel der RENUMBER-Funktion gesehen.

Eine Zusammenstellung der oben besprochenen Erweiterungen finden Sie in Bild 5.

Strubs und Interpretererweiterungen

Wollen Sie mit Strubs Programme für Interpretererweiterungen bearbeiten, dann sind einige weitere Dinge zu beachten. Entfernen Sie zunächst wie in Folge 3 beschrieben die Interpretererweiterung von Strubs.

Falls die Erweiterung, die Sie benutzen wollen, nicht in den Editor eingreift, sondern ihre neuen Befehle durch besondere Zeichen (meistens »!«) gekennzeichnet werden, dann ändern Sie wie bereits oben beschrieben die entsprechenden Erkennungszeichen, die Strubs benutzt.

Bei Erweiterungen wie Simon's Basic, die in den Editor eingreifen und die neuen Befehle wie der Basic-Interpreter durch eigene Tokens darstellen, ist es am einfachsten, den Strubs-Befehlen, deren Namen solche Befehle enthalten, neue Namen zu geben. Im Fall von Simon's Basic sind davon beispielsweise Strubs-Befehle wie »!REPEAT«, »!UNTIL« oder »!ELSE« etc. betroffen.

Dazu sind nur die Namen in den DATA-Zeilen 45272 bis 45274 zu ändern. Sie können die betroffenen Strubs-Befehle aber auch wie oben am Beispiel von »DEFMAKRO« beschrieben aus den Tokens zusammensetzen. Dabei ist aber zu berücksichtigen, daß die Tokens von Simon's Basic aus zwei Zeichen und nicht wie die normalen Tokens aus nur einem Zeichen bestehen.

Eine Liste der von Strubs benutzten Variablen bietet Bild 6. Dabei kennzeichnet das Zeichen »*« Zeilennummern, in denen eine Wertzuweisung an die Variable erfolgt.

Zum Abschluß noch einige Berichtigungen zu den ersten Folgen: Die abgedruckte Programmversion wird nicht wie behauptet mit einem Kaltstart, sondern mit normalem »END« (Zeile 40190) beendet. Bei den in Teil 2 auf S. 121 angeführten Beispielen für Markendefinition und Externdeklaration haben sich Fehler eingeschlichen. Die korrekte Form entnehmen Sie bitte Bild 7.

(Matthias Törk)

```
10 £X-AUSGEBEN:
20 PRINT "X:";X
30 RETURN

READY.

20 REM VEREINBARUNG:
30 !EXT:£MAPRO:740,£PLOT:50000

READY.
```

Bild 7. Berichtigung zur Folge 2, Seite 121

REISE DURCH DIE WU

TEIL 4

Nachdem in den ersten drei Folgen unseres Grafikkurses al Die hochauflösende Grafik des Commodore 64, die von u men einzusetzen. Die nötigen Hilfsmittel, wie zum Beispiel e

Sie können bisher Zeichen im Mehrfarbenmodus und (aber nicht gleichzeitig) mit veränderten Hintergrundfarben darstellen. Das Prinzip der Bit-Map ist Ihnen vertraut und Sie wissen, wie man dem Computer sagt, daß er nun seine Bildschirminformationen aus dieser Bit-Map holt. Sie können in diesem Modus die Farben bestimmen und schließlich auch Punkte exakt in die Bit-Map setzen. Wenn Ihnen der Inhalt der Bit-Map nicht mehr gefällt, können Sie sie löschen.

In dieser vierten Folge werden wir lernen, wie man die Speicher des Commodore 64 für Grafikanwendungen umkrempelt. Wir werden »Dornröschen« in mehreren Farben erleben und schließen diesen Teil der hochauflösenden Grafik mit einer kleinen Unterprogrammbibliothek ab. Der einleitenden Worte sind genug gesagt, Dornröschen wartet.

Wir krempeln den Commodore 64 um: Speicherveränderungen für hochauflösende Grafik

Wenn Sie als stolzer Besitzer eines C 64 früher auch mal ebenso stolzer Besitzer eines VC 20 waren, dann ist Ihnen sicherlich in wehmütiger Erinnerung, was Sie sehen, wenn Sie durch die POKE-Kommandos
POKE 51,255:POKE 52, 31:POKE 55,255:POKE 56,31
in der letzten Folge die Bit-Map vor dem Überschreiben durch Basic geschützt haben und dann mal mit PRINT FRE (1) nach dem freien Basic-Speicher fragten: Da zeigte sich: 6144 (ohne Programm)!

Geht das Ringen um jedes Byte nun wieder los? Wie soll denn in diese 6 KByte ein besseres Spiel mit Hochauflösungsgrafik — ganz zu schweigen von anspruchsvolleren Programmen, zum Beispiel einer Kurvendiskussion — hineinpassen? Nun, keine Sorge: Wozu haben wir im C 64 denn 64 KByte RAM? Wir müßten nur wissen, wie wir sie nutzen können.

Dazu sehen wir uns nochmal den VIC-II-Chip an. Im Gegensatz zur CPU (unserem Prozessor 6510), die über 16 Adressenleitungen verfügen kann, stehen beim VIC-II-Chip lediglich 14 zur Disposition. Während man also mit 16 Leitungen alle Adressen
von 0 bis 1111 1111 1111 1111 = 65535
ansteuern kann, ist bei 14 Leitungen ein Maximum
von 11 1111 1111 1111 = 16383 Adressen
möglich, also 16 KByte.

Der gesamte Speicherraum des C 64 ist in vier solche 16 KByte-Blöcke aufteilbar und wie wir wissen, blickt der VIC-II-Chip im Normalfall auf den ersten 16 KByte-Block (siehe Bild 1). Nun kann man dem VIC-II-Chip mitteilen, daß er seine Aufmerksamkeit auf die anderen Speicherviertel richten möge. Das erfordert die Mitarbeit des »Portiers« CIA 2 (siehe Folge 1). Er hat im Gebäude 56576 zwei Zimmer (Bits 0 und 1), aus denen dem VIC-II-Chip die Anweisungen gegeben werden, um welches Viertel unseres Speichers er sich kümmern soll. Auf welche Weise diese Bits den VIC-II-Chip-Zugriff regeln, sehen Sie aus Tabelle 1.

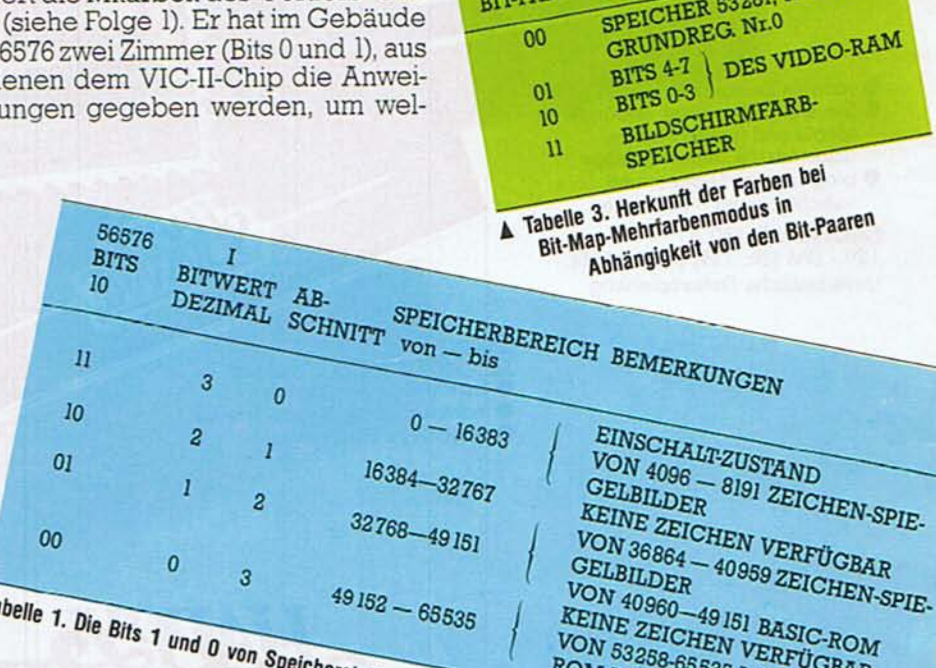
Commodore empfiehlt nun noch sicherzustellen, daß die zu dieser Abschnittsauswahl gehörigen Bits des Datenrichtungsregisters Port A im CIA 2 (Speicherstelle 56578) auf 1, also auf Ausgabe, gestellt werden. In allen Programmen, die ich bisher verwendet habe, wäre das eigentlich nicht nötig gewesen, jedenfalls habe ich nichts bemerkt, als ich das nicht getan habe.

Trotzdem gebe ich hier diese Empfehlung weiter, falls in Ihren Programmen diese Maßnahme notwendig wird. Falls Sie vergessen haben sollten (Folge 2), wie man Bits setzt oder löscht, hier die nötigen Programmzeilen dazu:

```
20 POKE 56576, (PEEK(56576) AND
252) OR I
30 POKE 56578, PEEK(56578) OR 3
```

I ist dabei der in Tabelle 1 gezeigte Dezimalwert der Bits 0 und 1.

Der VIC-II-Chip managt auch den Bildschirm. Im Einschaltzustand packt er den Bildschirmspeicher — wie wir schon wissen — in den Bereich 1024 und 2023. Wenn wir nun einen anderen 16 KByte-Abschnitt wählen, legt er den Bildschirm an die entsprechende Stelle dieses Abschnittes, also:
In Abschnitt 0: Bildschirm von 1024 bis 2023
in Abschnitt 1: Bildschirm von 16384



BIT-PAAR	FARBQUELLE
00	SPEICHER 53281, HINTERGRUNDREG. Nr.0
01	BITS 4-7 DES VIDEO-RAM
10	BITS 0-3 DES VIDEO-RAM
11	BILDSCHIRMFARBSPEICHER

▲ Tabelle 3. Herkunft der Farben bei Bit-Map-Mehrfarbenmodus in Abhängigkeit von den Bit-Paaren

56576 BITS 10	I BITWERT DEZIMAL	ABSCHNITT	SPEICHERBEREICH von — bis	BEMERKUNGEN
11	3	0	0 — 16383	EINSCHALTZUSTAND VON 4096 — 8191 ZEICHEN-SPIEGELBILDER
10	2	1	16384—32767	KEINE ZEICHEN VERFÜGBAR
01	1	2	32768—49151	VON 36864 — 40959 ZEICHEN-SPIEGELBILDER
00	0	3	49152 — 65535	VON 40960—49151 BASIC-ROM KEINE ZEICHEN VERFÜGBAR VON 53258-65535 I/O, ZEICHEN-ROM BETRIEBSSYSTEM

Tabelle 1. Die Bits 1 und 0 von Speicherstelle 56576 regeln den Zugriff des VIC-II-Chips auf den Speicher

DERWELT DER GRAFIK

Grundlagen geschaffen wurden, nähern wir uns endlich mit Riesenschritten unserem Ziel: s endlich aus ihrem Dornröschenschlaf geweckt wurde, gezielt in unseren Basic-Program- e kleine Bibliothek von Grafik-Unterprogrammen, werden in dieser Folge vorgestellt.

+ 1024 = 17408
bis 16384 + 2023 = 18407
in Abschnitt 2: Bildschirm von 32768 + 1024 = 33792
bis 32768 + 2023 = 34791
in Abschnitt 3: Bildschirm von 49152 + 1024 = 50176
bis 49152 + 2023 = 51175.

Damit brauchen wir uns aber nicht zufrieden geben.

Multivisio — unter 64 Bildschirm wählen

Im 16 KByte-Speicherabschnitt hat der Bildschirmspeicher ja 16mal Platz und wir können ihn ohne weiteres an eine andere Stelle schieben. Damit kehren wir nochmal zur schon oft besungenen Speicherstelle 53272 zurück. Die Bits 4 bis 7 geben dem VIC-II-Chip den Ort des Bildschirmspeichers an. Durch Verändern dieser 4 Bits können wir tatsächlich die 16 Bildschirme pro 16 KByte-Abschnitt einrichten. Der Zusammenhang zwischen den Bits 4 bis 7 von Adresse 53272 und dem Ort des Bildschirmspeichers im Abschnitt 0 (und entsprechend parallelverschoben in den anderen Abschnitten) ist in Tabelle 2 gezeigt.

SPEICHERSTELLE 53272 BITS: 7654	DEZIMALWERT (BITS 0-3 als 0 angenommen) W	BILDSCHIRMSTARTADRESSE (IM ABSCHNITT 0) DEZIMAL	HEX
0000	0	0	0
0001	16	1024	400
0010	32	2048	800
0011	48	3072	C00
0100	64	4096	1000
0101	80	5120	1400
0110	96	6144	1800
0111	112	7168	1C00
1000	128	8192	2000
1001	144	9216	2400
1010	160	10240	2800
1011	176	11264	2C00
1100	192	12288	3000
1101	208	13312	3400
1110	224	14336	3800
1111	240	15360	3C00

Tabelle 2. Zusammenhang zwischen den Bits 4 bis 7 von Speicher 53272 und dem Ort des Bildschirms in Abschnitt 0

Um die entsprechende Bitanordnung zu erreichen, müssen wir also eingeben:

```
40 POKE 53272, (PEEK(53272) AND 15) OR W
```

Dabei ist W der Dezimalwert der Bits 4 bis 7 aus Tabelle 2.

Das Betriebssystem muß auch noch erfahren, daß der Bildschirmspeicher verlegt worden ist. Man kann es ihm mitteilen, indem man die Pagenummer der Bildschirmstartadresse in Speicherstelle 648 einPOKEd. Also ist zum Beispiel der normale Inhalt von Speicher 648: 1024/256 = 4. Auf Page 4 beginnt der Bildschirm ja im Einschaltzustand. Die Pagenummer ergibt sich aus I und W (siehe Tabelle 1 und 2) durch folgende Rechnung:

```
50 P = (W/16*1024 + 16384*(3-I))/256
```

und wird dann eingePOKEd:

```
55 POKE 648,P
```

So! Jetzt können wir theoretisch 64 Bildschirme erzeugen. Um uns das in der Praxis mal anzusehen, ergänzen wir die bisher verwendeten Programmzeilen (die Sie hoffentlich noch nicht mit RUN gestartet haben, das wäre nämlich mit etwas Glück eine gute Methode, den Rechner abstürzen zu lassen!) noch um folgendes:

```
10INPUT"I,W";I,W:PRINT CHR$(147)
60 PRINT CHR$(147)I,W:END
65 PRINT CHR$(147)
70 POKE 56576,151:POKE 56578,63:
POKE 53272,21:POKE 648,4
80 I=3:W=16:PRINT CHR$(147)I,W
90 END
```

Bevor Sie das starten, sollten Sie bitten daran denken, daß einige I,W-Kombinationen den Computer zum totalen Black-Out führen: Zum Beispiel I=3,W=0 legt den Bildschirmstart direkt in die Zeropage, ist also nicht empfehlenswert. I=3,W=32 zerstört unser Programm, das genau bei 2048 anfängt. Am besten speichern Sie das Programm vor dem Starten ab.

Jeweils nach RUN und Eingabe der Werte I und W, zum Beispiel I=3, W=48, wird der Bildschirm gelöscht und in der obersten Zeile wird der I- und der W-Wert angegeben. Danach meldet sich READY und ein Cursor. Jetzt befinden wir uns im neuen Bildschirm (unser Beispiel also 3072 bis 4071) und können damit herumexperimentieren.

Bildschirm-Experimente

Wenn wir jetzt (falls keine Programmänderung in der Zwischenzeit durchgeführt wurde) CONT eingeben, wird der Ursprungszustand (Bildschirm bei 1024) wieder hergestellt und dies durch die Angabe der I- und W-Werte 3 und 16 angezeigt.

Wenn Sie mit diesem Programm in den Bereich 4096 bis 8191 vorstoßen, werden Sie feststellen, daß hier kein normaler Bildschirm möglich ist. Hier stören die mehrfach schon beschworenen Geisterbilder des Zeichen-ROM, die in diesem Bereich liegen. Es kann sogar passieren, daß der Rechner nach der Eingabe von CONT nur noch SYNTAX ERRORs meldet und nicht mehr in den Normalzustand zurückzuführen ist. Ab 8192 bis 15360 (jeweils Start des Bildschirmes) kann man wieder ohne Störung Bildschirme einrichten. Wenn Sie jetzt mal I=2 und verschiedene W-Werte versuchen, sehen Sie nur Nonsens oder gar nichts auf dem Bildschirm, dasselbe geschieht bei I=0.

Das ist wieder eine Besonderheit des VIC-II-Chip. Er ist so strukturiert, daß der (im Normalfall) in diesen beiden Abschnitten keinen Zugang zu den normalen Zeichenspeichern hat. Dafür gibt es in Abschnitt 1 (I = 2) keine Störung durch die Zeichen-Geisterbilder, ebenso in Abschnitt 3 (I = 0). In Abschnitt 2 (I = 1) begegnen wir zwischen 36864 und 40959 wieder den hier ein zweites Mal vorhandenen Zeichen-Gespenstern. Unterhalb von 36864 läßt sich der neue Bildschirm gut verwenden.

Der verborgene Speicher — RAM-Bereiche unter dem ROM

Ein Problem stellt sich hier und auch im obersten Abschnitt aber noch auf andere Weise: Wenn wir den Bildschirm zum Beispiel mit I = 1 und W = 128 nach 40960 legen (tun Sie es bitte nicht!), dann erhalten wir bei jeder Eingabe nur noch »SYNTAX ERRORs« und können den Computer nur durch Aus- und Einschalten wieder zu normaler Tätigkeit bewegen. Was ist da los? Die Erklärung ist, daß von 40960 bis 49151 das Basic-ROM und von 57344 bis 65535 das Betriebssystem dem RAM überlagert sind. Wenn man in diese Regionen hineinPOKEd, landet die Information natürlich im darunterliegenden RAM. Das was auf dem Bildschirm erscheint wenn wir aus dem Programm-Modus aussteigen, ist allerdings — leider — der Inhalt des darüberliegenden ROM. Ersetzen Sie aber mal das »END« in Zeile 60 durch die folgenden Zeilen:

```
60 PRINT CHR$(147)I,W:PRINT"
BILDSCHIRM LIEGT UNTER DEM
ROM«
65 GET A$:IF A$ = "" THEN 65
```

Siehe da! Es funktioniert also im Programm-Modus (zum Beispiel mit I = 1 und W = 128). Wir können daher unter das Basic-ROM auf diese Weise acht Bildschirme legen. Unter das Betriebssystem lassen sich so auch Bildschirme legen, nur können wir hier den Text nicht lesen, weil der VIC-II-Chip — wie gesagt — hier keinen Zugriff zum Zeichen-ROM hat. So legt zum Beispiel I = 0 und W = 240 den Bildschirm nach 64512, was leicht nachprüfbar ist durch die Zeile:

```
62 POKE 65000, 1:POKE 55784,1.
```

Damit wenden wir uns nun dem kritischen Bereich zwischen 53248 und 57343 zu. Hier liegen ja das Zeichen-ROM und die Ein- und Ausgabebausteine. Normalerweise — wie man auch durch unsere POKE in diesen Bereich erkennen kann — sind hier die Ein- und Ausgabe-Bausteine eingeschaltet. Wenn wir hierher Bildschirme legen, kann alles mögliche passieren, weil wir Register des VIC-II-Chip, des SID und CIAs beeinflussen. Hier sollte man mit viel Vorsicht und gegebenenfalls nur in Maschinensprache operieren.

Was wir durch die Programmzeile 62 noch erkennen können: Das Bildschirmfarben-RAM verschiebt sich nicht, egal, welches Speicherviertel wir wählen und wohin wir den Bildschirm auch legen: Das Farb-RAM liegt immer von 55296 bis 56295.

Wohin mit der Bit-Map?

Nun aber zum großen Speicherfresser: Zur Bit-Map. Mit ihren 8000 Byte paßt sie im Prinzip achtmal in unseren Computer. Im ersten Viertel (0 bis 16383) haben wir sie schon gehabt und das als unbefriedigend empfunden. Nun wollen wir uns andere Möglichkeiten ansehen und dabei noch bedenken, daß wir auf den normalen Zeichensatz verzichten können (wir stellen ja hochauflösende Grafik dar!). Zu diesem Zweck werden wir das bisher verwendete Bildschirmtestprogramm um einen Hochauflösungsteil erweitern (Listing 1). Um die leidige Eintipperei minimal zu halten, wurde auf Schönheit und erläuternde REM-

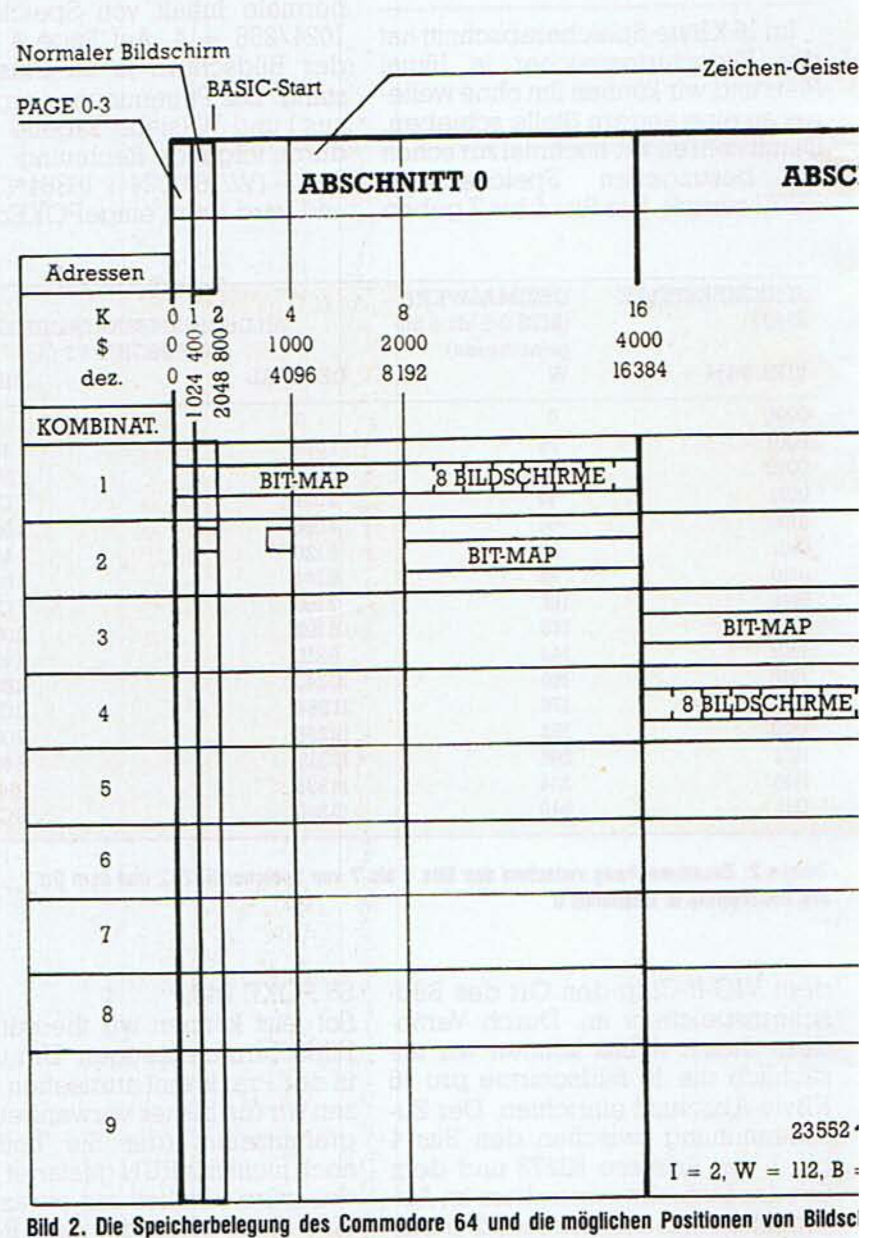


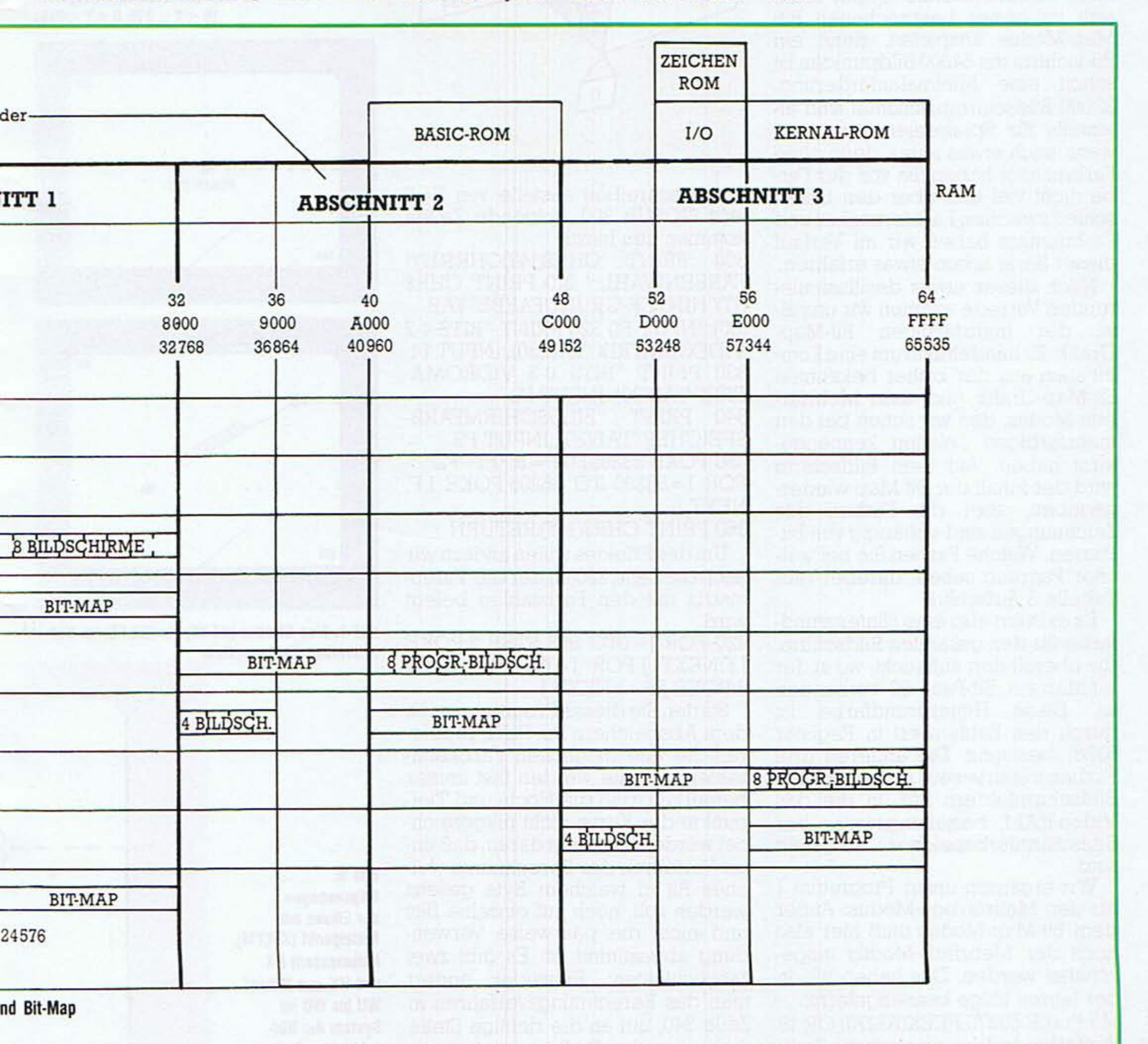
Bild 2. Die Speicherbelegung des Commodore 64 und die möglichen Positionen von Bildsch

Zeilen verzichtet. Der Hochauflösungsteil stimmt weitgehend mit dem Programm aus der letzten Folge überein. Geben Sie also jetzt das Listing 1 ein, speichern Sie es möglichst gleich ab und probieren Sie es aus.

In Bild 2 sind alle möglichen Positionen der Bit-Map und des Bildschirmes angegeben.

Wie man sehen kann, scheiden die Kombinationen Nummer 1 und Nummer 7 von vornherein aus, weil wir mit dem Löschen der Bit-Map auch das Lebenslicht unseres Rechners ausblasen. Die Kombination Nummer 2 kennen wir schon: So haben wir in der letzten Folge hochauflösende Grafik betrieben und waren enttäuscht über den geringen verbliebenen Basic-Speicher. Bei Nummer 5 funken uns die Zeichen-Spiegelbilder in die Bit-Map, diese Kombination scheidet also auch aus. Nett sieht es aus, wenn wir die Kombinationen Nummer 6 und Nummer 8 testen. Hier machen sich die ROM-Inhalte grafisch zwar ganz interessant aus, aber mit dem Sinn unserer hochauflösenden Grafik hat das nichts mehr zu tun. Für uns brauchbar sind die Positionen im Abschnitt 1: Kombinationen Nummer 3 und Nummer 4. Ein Maximum an Basic-Speicher findet man bei der letzten gezeigten Kombination Nummer 9, wo ab 23552 der Bildschirm und ab 24576 die Bit-Map liegen. Wenn Sie den Basic-Speicher für diese Anordnung schützen, mit POKE 55,0:POKE 56,92:POKE 52,92 und den Computer dann mit PRINT FRE(I) nach dem freien Speicherplatz fragen, dann erhalten Sie als Antwort immerhin satte 21501 freie Byte.

Der Idealfall wäre es, wenn man die Bit-Map unter das ROM legen könnte (Kombinationen 6 oder 8). Das geht natürlich auch! Von Basic aus wird ein Programm dann allerdings noch langsamer, weil man für jeden Punkt ähnliche Operationen vornehmen müßte, wie wir sie in der zweiten Folge beim Kopieren des Zeichen-ROM ins RAM verwendet haben. Auch so ist die ganze Hochauflösungsgrafik schon ziemlich langsam. Wir werden aber in kommenden Folgen einige Routinen in Maschinensprache kennenlernen, die uns mehr Möglichkeiten eröffnen.



...nd Bit-Map

Hochauflösende Grafik in Farbe: Kann das der C 64 überhaupt? Die Antwort lautet Ja und Nein. Ja, weil der bislang von uns verwendete Bit-Map-Modus anstelle der zwei bisher benutzten auch mit vier Farben ablaufen kann. Nein, weil die Punktauflösung dann eigentlich nicht mehr die Bezeichnung »hochauflösend« verdient. Die horizontale Auflösung geschieht hier nämlich nur noch in 160 Positionen anstelle der

Die Grafik bekennt Farbe: Der Bit-Map-Mehrfarben-Modus

bisher ansprechbaren 320. Meine persönliche Meinung dazu ist, ernsthafte hochauflösende Grafik sollte sich im bisher besprochenen Bit-Map-Modus abspielen, denn ein Bildschirm mit 64000 Bildpunkten ist schon eine Minimalanforderung. 32000 Bildschirmpositionen sind allenfalls für Spielereien ganz nett, wenn auch etwas teuer, denn ohne Farbmonitor haben Sie von der Farbe nicht viel und über den Unterschied zwischen Farbfernseher und Farbmonitor haben wir im Verlauf dieser Serie schon etwas erfahren.

Nach dieser etwas desillusionierenden Vorrede widmen wir uns also der mehrfarbigen Bit-Map-Grafik. Es handelt sich um eine Kombination aus der bisher bekannten Bit-Map-Grafik und dem Mehrfarben-Modus, den wir schon bei den mehrfarbigen Zeichen kennengelernt haben. Auf dem Bildschirm wird der Inhalt der Bit-Map wiedergegeben, aber die Farben der Zeichnungen sind abhängig von Bit-Paaren. Welche Farben Sie bei welcher Paarung sehen, darüber gibt Tabelle 3 Aufschluß.

Es existiert also eine Hintergrundfarbe für den gesamten Bildschirm, die überall dort auftaucht, wo in der Bit-Map ein Bit-Paar 00 vorhanden ist. Diese Hintergrundfarbe ist durch den Zahlenwert in Register 53281 bestimmt. Die anderen drei Farben treten jeweils in den 8 x 8-Bit-Bildschirmfeldern auf, in die das Video-RAM, beziehungsweise der Bildschirmfarbspeicher aufgeteilt sind.

Wir ergänzen unser Programm 1 für den Mehrfarben-Modus: Außer dem Bit-Map-Modus muß hier also noch der Mehrfarb-Modus eingeschaltet werden. Das haben wir in der letzten Folge kennengelernt:

145 POKE 53270,PEEK(53270) OR 16

Weiterhin ändern wir noch die Zeile

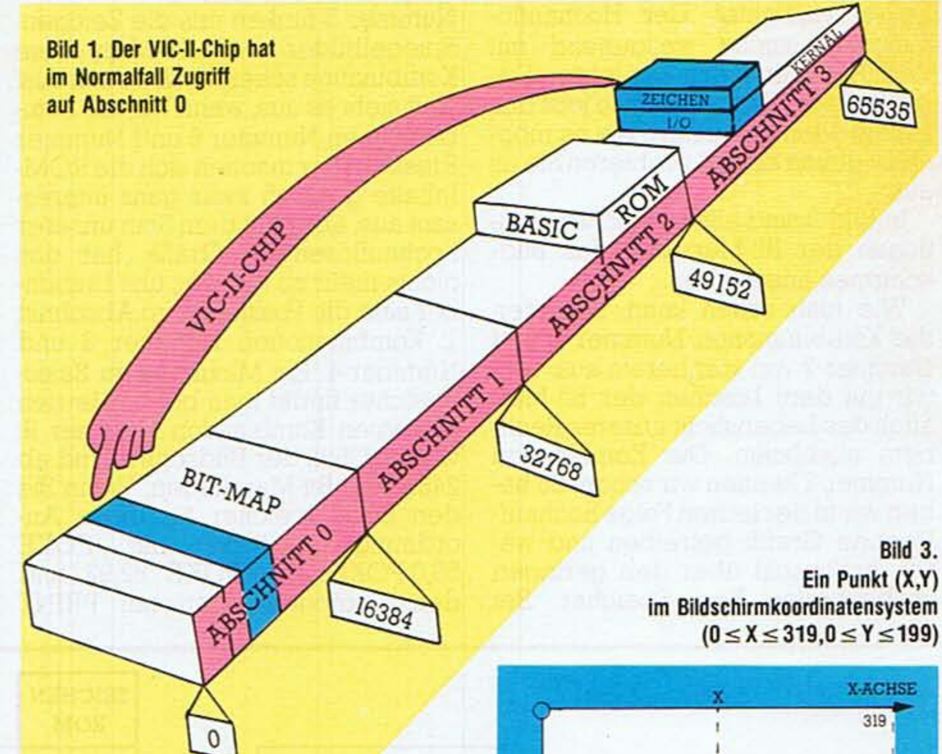


Bild 1. Der VIC-II-Chip hat im Normalfall Zugriff auf Abschnitt 0

5 und schreiben anstelle von F=6 jetzt GOSUB 300. Folgende Zeilen kommen neu hinzu:

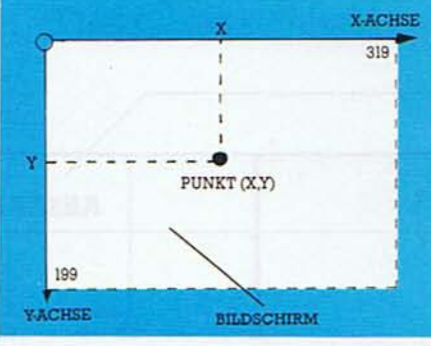
```
300 PRINT CHR$(147)CHR$(17)"
FARBENWAHL:" 310 PRINT CHR$
(17)"HINTER-GRUNDFARBE"TAB
(30);:INPUT F0 320 PRINT "BITS 4-7
VIDEOMATRIX"TAB(30);:INPUT F1
330 PRINT "BITS 0-3 VIDEOMA-
TRIX"TAB(30);:INPUT F2
340 PRINT "BILDSCHIRMFARB-
SPEICHER"TAB(30);:INPUT F3
350 POKE 53281,F0:F=16*F1+F2:
FOR I=55296 TO 56295:POKE I,F:
NEXT I
360 PRINT CHR$(147):RETURN
```

Um des Effektes willen ändern wir noch die Zeile 220, in der die Videomatrix mit den Farbzahlen belegt wird:

```
220 FOR J=0TO 998 STEP 2:POKE
J,F:NEXT J:FOR J=1TO 999 STEP
2:POKE J,F+1:NEXT J
```

Starten Sie dieses Programm nach dem Abspeichern mit RUN, probieren Sie alle möglichen Farbkennzahlen aus. Sie werden fast immer bemerken, daß die Hoch- und Tiefpunkte der Kurve nicht mitgezeichnet werden. Das liegt daran, daß unter Verfahren der Berechnung, welches Bit in welchem Byte gesetzt werden soll, noch auf einzelne Bits und nicht die paarweise Verwendung abgestimmt ist. Es gibt zwei Möglichkeiten: Entweder ändert man das Berechnungsverfahren in Zeile 240, um an die richtige Stelle die passenden Bit-Paare zu bekom-

Bild 3. Ein Punkt (X,Y) im Bildschirmkoordinatensystem ($0 \leq X \leq 319, 0 \leq Y \leq 199$)



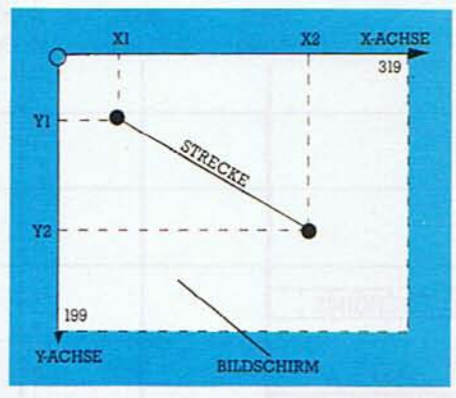
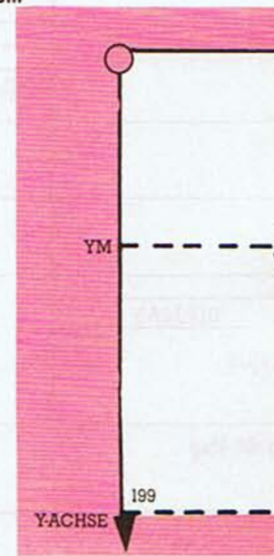


Bild 4: Eine Strecke (X1,Y1) bis (X2,Y2) im Bildschirmkoordinatensystem

Bild 5: Ellipsenbogen der Ellipse mit Mittelpunkt (XM,YM), Halbmessern HX und HY, von Winkel WU bis WO im System der Bildschirmkoordinaten



men, oder wir ändern die Programmzeile 230 zu:
230 FOR I=0 TO 319 STEP 2:Y=FNA(X)

Das ist zwar mal wieder etwas primitiv, aber — wie gesagt — sind meine Ambitionen zur Mehrfarben-»Hochauflösung« sowieso nicht so stark. Wer Lust hat, kann sich ja gerne mal mit der korrekten Weise der neuen Berechnung herumschlagen. Das Wissen, diese Aufgabe zu bewältigen, haben Sie jetzt. Auf dem Schwarzweiß-Monitor sieht zum Beispiel folgende Zahlenkombination ganz gut aus:
Hintergrundfarbe: 7
Bits 4 bis 7: 5
Bits 0 bis 3: 1
Bildschirmspeicher: 0
Außerdem natürlich noch: I=2, W=112, B=1.

Mit dem hier folgenden Abschnitt soll zunächst einmal die hochauflösende Grafik beiseite gelegt werden. Dornröschen ist erwacht und genesen, allerdings noch nicht voll bei Kräften. Das wird in einer späteren Folge noch anders werden. Davor wollen wir aber noch weitere Grafik-Besonderheiten des C 64 behandeln. An dieser Stelle wollen wir jedoch einen Zwischenhalt einlegen und eine kleine Sammlung von Basic-Unterprogrammen zur Grafik-Programmierung vorstellen. In Listing 2 sind diese Grafik-Unterprogramme, in Listing 3 ein Beispiel-Aufrufprogramm abgedruckt. Beim Eintippen beider Programme können Sie die REM-Zeilen ohne Schaden weglassen.

Erläuterungen zu Listing 2 (Zeilenbereich 49990 bis 51500)

◆Sprungtabelle:
Das ist in Basic im allgemeinen nicht üblich, sondern wird häufiger bei Maschinensprache-Programmen verwendet. Trotzdem hat es auch hier seine Vorteile. Es kann ja sein, daß Sie einige Änderungen oder Ergänzungen in den Unterprogrammen vornehmen wollen. Sie müßten dann auch immer die Adressen in den jeweils aufzurufenden Hauptprogrammen umschreiben. Mit der Sprungtabelle ist das nicht mehr nötig, denn die GOSUB-Adressen im Hauptprogramm bleiben unverändert, nur die neuen GOTO-Adressen im Unterprogramm sind einzusetzen.

◆ HIRES an
Hier machen wir uns die Erkenntnisse dieser Folge zunutze und legen den Bildschirm nach 23552 und die Bit-Map nach 24576. Beides müssen wir — wie gehabt — vor dem Überschreiben durch Basic schützen mit:
POKE 52,92:POKE 56,92.
Am besten packt man diese POKE-Befehle gleich in die ersten Zeilen des aufrufenden Hauptprogramms.

◆Bit-Map-Löschen
Hierzu gibt es nichts mehr zu sagen, außer, daß I als Laufvariable dient.

◆Farbgebung
Bevor dieses Unterprogramm aufgerufen wird, müssen im Hauptprogramm
F1 = Zeichenfarbe und
F2 = Hintergrundfarbe
definiert sein. An Variablen treten noch auf:
F = Farbcodezahle auf F1 und F2
I = Laufvariable

◆HIRES aus
Dieses Programm stellt die ursprüngliche Speicherorganisation wieder her (Bildschirm- und Zeichenspeicher) und schaltet in den Normalmodus zurück.

◆Punkt setzen
Auch hier müssen vor dem Aufruf des Unterprogramms im Hauptspeicher die
Punktkoordinaten X,Y
definiert sein (siehe Bild 3) sowie die
L — Löschmarke
Wenn L = 0 ist, wird der Punkt gesetzt (Zeile 50930), so, wie wir das schon kennen. Ist L = 1, dann wird ein vorhandener Punkt gelöscht (Zeile 50920).
Die Zeile 50905 achtet darauf, daß keine Punktkoordinate außerhalb des Bildschirms liegt, was unter Umständen ein Aussteigen mit Fehlermeldung im Hochauflösungsverfahren zur Folge hätte. Das ist hier zwar nicht so tragisch, weil man durch Eingeben von GOTO 50030 »RETURN« schnell wieder in den Normalmodus gelangen kann (eventuell muß vorher noch »SHIFT« + »CLR/HOME« gedrückt werden). Trotzdem ist es dumm, wenn inmitten all dieser zeitraubenden Grafiktätigkeiten auch noch der Rechner aussteigt. Eine Grenzüberschreitung der Koordinaten ist um so leichter möglich, als die Punkt-Routine von allen folgenden Unterprogrammen aufgerufen wird. Außer X,Y und L tauchen noch die Variablen BY und BI auf, die wir schon kennengelernt haben als das Byte, in dem ein Bit zu setzen oder zu löschen ist.

◆Punkt löschen
Hier geschieht nichts anderes, als die Löschmarke L auf 1 zu setzen und dann in die Punkt-Setz-Routine zu springen. Deswegen gilt für dieses Unterprogramm dasselbe wie für das vorangegangene.

◆Strecke zeichnen
Vor dem Aufruf müssen dem Rechner schon
der Startpunkt (X1,Y1) und
der Endpunkt (X2,Y2) der Strecke
bekannt gemacht sein (siehe Bild 4).
Den mathematisch Versierten wird es bei der Betrachtung der Zeilen 51120 beziehungsweise 51160 schon aufgefallen sein, daß zur Berechnung der Punkte, aus denen sich die Strecke zusammensetzt, die sogenannte 2-Punkte-Form der Geradengleichung verwendet wurde:

$$\frac{Y-Y1}{X-X1} = \frac{Y2-Y1}{X2-X1}$$

Den mit Mathematik nicht so vertrauten Lesern sei gesagt, daß es sich um eine Formel aus der sogenannten analytischen Geometrie handelt. Das ist ein Gebiet der Mathematik, das für die Grafik auf Computern eine nicht unerhebliche Rolle spielt.
Die Punkte (X1, Y1) und (X2, Y2) dürfen auch außerhalb des Bildschirmsystems liegen. Im ersten Teil des Unterprogramms dient die Übergabevariable X auch gleichzeitig als Laufvariable während Y berechnet wird. Wenn allerdings der Absolutbetrag von X2-X1 kleiner als 5 wird, verkehren sich die Verhältnisse: Y wird Laufvariable und X berechnet. Das beschleunigt das Zeichnen von Senkrechten und verhindert außerdem eine Division durch Null. Der Wert von 5 ist dabei ziemlich willkürlich gewählt. L ist wieder die Löschmarke.

◆Strecke löschen
Es gilt dasselbe wie für das Unterprogramm Strecke zeichnen, nur daß hier wieder die Löschmarke gesetzt wird.

◆ **Ellipse zeichnen**

Vor dem Aufruf müssen folgende Werte schon definiert sein (siehe auch Bild 5):

(XM,YM) = Mittelpunktkoordinaten
HX = Halbmesser in X-Richtung
HY = Halbmesser in Y-Richtung
WU,WO = Der zu zeichnende Ellipsenbogen beginnt beim Winkel WU und endet beim Winkel WO (Gradmaß)

Eine volle Ellipse wird also gezeichnet, wenn WU = 0 und WO =

```
1 REM ***************************
2 REM *     PROGRAMM  1          *
3 REM ***************************
5 F=6:DEFFNA(X)=50*SIN(X/30)+100
10 PRINTCHR$(147)CHR$(17)"EINGABE DER WERTE"CHR$(17)
20 PRINTCHR$(17)"I(SIEHE TAB.1) ABSCHNITT-KENNZIFFER"CHR$(17)
30 PRINT"W(SIEHE TAB.2) BILDSCHIRM-KENNZIFFER"CHR$(17)
40 PRINT"B=0,BIT-MAP IM UNTEREN ABSCHNITT-BEREICH"
50 PRINT" =1,BIT-MAP IM OBEREN ABSCHNITT-BEREICH"CHR$(17)
60 PRINTCHR$(17):INPUT"I,W,B=";I,W,B:A1=3-I:A2=W/16*1024+A1*16384:A3=A1*16384+B*
8192
70 P=A2/256:PRINTCHR$(17)CHR$(17)"MIT DIESEN EINGABEN HABEN SIE"
80 PRINT"DEN ABSCHNITT "A1" GEWAEHLT."
90 PRINT"IHR BILDSCHIRM STARTET BEI "A2
100 PRINT"UND IHRE BIT-MAP BEI "A3
110 PRINTCHR$(17)"IST DAS SO IN ORDNUNG?(J/N)"
120 GETA$:IFA$=""THEN120
130 IFA$="N"THEN10
140 PRINTCHR$(147)
145 REM **** NEUER SPEICHER ****
150 POKE56576,(PEEK(56576)AND252)ORI
160 POKE56578,PEEK(56578)OR3
170 POKE53272,(PEEK(53272)AND15)ORW
180 POKE648,P
190 POKE53265,PEEK(53265)OR32
200 POKE53272,PEEK(53272)OR(8*B)
210 FORJ=0TO7999:POKEA3+J,0:NEXTJ
220 FORJ=0TO999:POKEA2+J,F:NEXTJ
230 FORX=0TO319:Y=FNA(X)
240 BY=(XAND504)+40*(YAND248)+(YAND7):BI=7-(XAND7)
250 POKEA3+BY,PEEK(A3+BY)OR(2↑BI):NEXTX
260 GETA$:IFA$=""THEN260
265 REM **** ALTER SPEICHER ****
270 POKE53272,21:POKE53265,27:POKE648,4:POKE56578,63:POKE56576,151
280 PRINTCHR$(147):END

READY.
```

Listing 1. Testprogramm zur Bildschirmlokalisierung

```
49990 REM -------------------
49991 REM --UNTERPROGRAMME-
49992 REM -------------------
49993 REM
49996 REM -------------------
49997 REM --SPRUNGTABELLE--
49998 REM -------------------
49999 REM
50000 GOTO50500:REM HIRES AN
50010 GOTO50600:REM BIT-MAP-LOESCHEN
50020 GOTO50700:REM FARBGEBUNG
50030 GOTO50800:REM HIRES AUS
50040 GOTO50900:REM PUNKT SETZEN
50050 GOTO51000:REM PUNKT LOESCHEN
50060 GOTO51100:REM STRECKE ZEICHNEN
50070 GOTO51200:REM STRECKE LOESCHEN
50080 GOTO51300:REM ELLIPSE ZEICHNEN
50090 GOTO51400:REM ELLIPSE LOESCHEN
50100 GOTO51500:REM KOMBINIERTES HIRES AN
50490 REM
50491 REM -------------------
50492 REM -----HIRES AN------
50493 REM -------------------
50494 REM
50500 POKE56576,(PEEK(56576)AND252)OR2
50510 POKE56578,PEEK(56578)OR3
50520 POKE53272,120:POKE648,92
50530 POKE53265,PEEK(53265)OR32
50540 RETURN
50590 REM
50591 REM -------------------
50592 REM -BIT-MAP-LOESCHEN--
50593 REM -------------------
50594 REM
50600 FORI=24576TO32575:POKEI,0:NEXTI
50610 RETURN
50690 REM
50691 REM --------------------
50692 REM -----FARBGEBUNG-----
50693 REM --------------------
50694 REM
50700 F=16*F1+F2
50710 FORI=23552TO24551:POKEI,F:NEXTI
50720 RETURN
50790 REM
50791 REM --------------------
50792 REM ------HIRES AUS-------
50793 REM --------------------
50794 REM
50800 POKE53265,27:POKE53272,21:POKE648,4
50810 POKE56578,63:POKE56576,151
50820 RETURN
50890 REM
50891 REM ---------------------
50892 REM -----PUNKT SETZEN-----
50893 REM ---------------------
50894 REM
50900 L=0
50905 IFX<0ORX>319ORY<0ORY>199THEN50940
50910 BY=(XAND504)+40*(YAND248)+(YAND7):BI=7-(XAND7)
50920 IFL=1THENPOKE24576+BY,PEEK(24576+BY)ANDNOT(2↑BI):GOTO50940
50930 POKE24576+BY,PEEK(24576+BY)OR(2↑BI)
50940 RETURN
50990 REM
50991 REM --------------------
50992 REM -----PUNKT LOESCHEN----
50993 REM --------------------
50994 REM
51000 L=1:GOTO50905
51090 REM
51091 REM --------------------
51092 REM ----STRECKE ZEICHNEN----
51093 REM --------------------
51094 REM
51100 L=0
51110 IFABS(X2-X1)<5THEN51150
51115 FORX=X1TOX2STEP(X2-X1)/319
51120 Y=(Y2-Y1)/(X2-X1)*(X-X1)+Y1
51130 GOSUB50905:NEXTX
51140 RETURN
51150 FORY=Y1TOY2STEP(Y2-Y1)/199
51160 X=(X2-X1)/(Y2-Y1)*(Y-Y1)+X1
51170 GOSUB50905:NEXTY
51180 RETURN
51190 REM
51191 REM --------------------
51192 REM ----STRECKE LOESCHEN-----
51193 REM --------------------
51194 REM
51200 L=1:GOTO51110
51290 REM
51291 REM --------------------
51292 REM -----ELLIPSE ZEICHNEN-----
51293 REM --------------------
51294 REM
51300 L=0
51310 FORW=WUTOWO:WB=W*π/180
51320 X=XM+HX*COS(WB):Y=YM+HY*SIN(WB)
51330 GOSUB50905
51340 NEXTW:RETURN
51390 REM
51391 REM --------------------
51392 REM -----ELLIPSE LOESCHEN-----
51393 REM --------------------
51394 REM
51400 L=1:GOTO51310
51490 REM
51491 REM --------------------
51492 REM --KOMBINIERTES HIRES AN---
51493 REM --------------------
51494 REM
51500 GOSUB50000:GOSUB50010:GOTO50020
```

Listing 2. Unterprogramme zur hochauflösenden Grafik

```
1 REM ******************************
2 REM *  GRAFIK TEST PROGRAMM VON   *
3 REM *  H.PONNATH 1984 VERBROCHEN  *
4 REM ******************************
5 POKE52,92:POKE56,92:DEFFNA(X)=50*SIN(X/30)+100
7 REM ******************************
8 REM *     MENUE-GUTEN APPETIT     *
9 REM ******************************
10 PRINTCHR$(147)CHR$(17)"UNTERPROGRAMME GRAFIK TEST"CHR$(17)
20 PRINTTAB(2)"NUR HIRES AN"TAB(25)"1"
30 PRINTTAB(2)"NUR HIRES AUS"TAB(25)"2"
40 PRINTTAB(2)"BIT MAP LOESCHEN"TAB(25)"3"
50 PRINTTAB(2)"FARBGEBUNG"TAB(25)"4"
60 PRINTTAB(2)"KOMBINATION"TAB(25)"5"
62 PRINTTAB(2)"PUNKTE SETZEN"TAB(25)"6"
64 PRINTTAB(2)"PUNKTE LOESCHEN"TAB(25)"7"
66 PRINTTAB(2)"STRECKE ZEICHNEN"TAB(25)"8"
68 PRINTTAB(2)"STRECKE LOESCHEN"TAB(25)"9"
70 PRINTTAB(2)"ELLIPSE ZEICHNEN"TAB(25)"A"
72 PRINTTAB(2)"ELLIPSE LOESCHEN"TAB(25)"B"
74 PRINTTAB(2)"DEMONSTRATION"TAB(25)"C"
76 PRINTTAB(1)"MENUE"TAB(25)"M"
78 PRINTTAB(1)"AUSSTEIGEN"TAB(25)"←"
80 GETA$:IFA$=""THEN80
90 IFA$="←"THENEND
92 IFA$="A"THENA$="10"
94 IFA$="B"THENA$="11"
96 IFA$="C"THENA$="12"
98 IFA$="M"THEN10
100 ONVAL(A$)GOSUB50000,50030,50010,200,300,400,500,600,700,800,900,1000
110 GOTO80
190 REM ***************************
191 REM *     FARBGEBUNG          *
192 REM ***************************
200 INPUT"ZEICHENFARBE,HINTERGRUNDFARBE=";F1,F2:GOTO50020
290 REM ***************************
291 REM * FARBE FUER KOMBINATION  *
292 REM ***************************
300 INPUT"ZEICHENFARBE,HINTERGRUNDFARBE=";F1,F2:GOTO50100
390 REM ***************************
391 REM * BEISPIEL PUNKTE SETZEN  *
392 REM ***************************
400 GOSUB50000:FORX=0TO319:Y=FNA(X):GOSUB50040:NEXTX
410 RETURN
490 REM ***************************
491 REM * BEISPIEL PUNKTE LOESCHEN*
492 REM ***************************
500 GOSUB50000:FORX=20TO250:Y=FNA(X):GOSUB50050:NEXTX
510 RETURN
590 REM ***************************
591 REM * AUFRUFPROGRAMM FUER     *
592 REM * STRECKE ZEICHNEN        *
593 REM ***************************
600 PRINT"ЯSTRECKE■ VON (X1,Y1) BIS (X2,Y2)":INPUT"X1,Y1,X2,Y2=";X1,Y1,X2,Y2
610 GOSUB50000:GOTO50060
690 REM ***************************
691 REM * AUFRUFPROGRAMM FUER     *
692 REM * STRECKE LOESCHEN        *
693 REM ***************************
700 PRINT"ЯSTRECKE■ VON (X1,Y1) BIS (X2,Y2)":INPUT"X1,Y1,X2,Y2=";X1,Y1,X2,Y2
710 GOSUB50000:GOTO50070
790 REM ***************************
791 REM * AUFRUFPROGRAMM FUER     *
792 REM * ELLIPSE ZEICHNEN        *
793 REM ***************************
800 PRINT"ЯELLIPSE■ MIT MITTELPUNKT (XM,YM),":PRINTTAB(1)"HALBMESSERN HX UND HY"
810 PRINTTAB(1)"ZEICHNEN VON WINKEL WU":PRINTTAB(1)"BIS WINKEL WO (GRADMASS)"
820 INPUT"XM,YM,HX,HY,WU,WO=";XM,YM,HX,HY,WU,WO:GOSUB50000:GOTO50080
890 REM ***************************
891 REM * AUFRUFPROGRAMM FUER     *
892 REM * ELLIPSE LOESCHEN        *
893 REM ***************************
900 PRINT"ЯELLIPSE■ MIT MITTELPUNKT (XM,YM),":PRINTTAB(1)"HALBMESSERN HX UND HY"
910 PRINTTAB(1)"LOESCHEN VON WINKEL WU":PRINTTAB(1)"BIS WINKEL WO (GRADMASS)"
920 INPUT"XM,YM,HX,HY,WU,WO=";XM,YM,HX,HY,WU,WO:GOSUB50000:GOTO50090
990 REM ***************************
991 REM * AUFRUFPROGRAMM FUER     *
992 REM * DEMONSTRATION           *
993 REM ***************************
1000 GOSUB50030:PRINTCHR$(147):FORI=1TO10:PRINTCHR$(17);:NEXTI
1010 PRINTTAB(8)"BITTE ETWAS GEDULD":PRINTTAB(5)"DIE BIT-MAP WIRD GELOESCHT"
1020 GOSUB50010:PRINTTAB(5)"UND MIT FARBE VERSEHEN":FORI=1TO1000:NEXTI
1030 F1=7:F2=6:GOSUB50000:GOSUB50020:FORI=1TO500:NEXT:GOSUB50030
1040 PRINTCHR$(147):FORI=1TO10:PRINTCHR$(17);:NEXTI
1050 PRINTTAB(5)"ZEICHNEN VON STRECKEN":FORI=1TO500:NEXTI
1060 GOSUB50000:FORI=0TO12
1070 X1=30+I*10:Y1=180-I*14.17:X2=150+I*13.3:Y2=10+I*14.583
1080 GOSUB50060:NEXTI:FORK=0TO9:F1=K:F2=K+1:GOSUB50020:FORJ=1TO500:NEXTJ:NEXTK
1090 GOSUB50030:PRINT:PRINTTAB(5)"STRECKEN LOESCHEN"
1100 FORI=1TO500:NEXTI:GOSUB50000:X1=30:Y1=180:X2=150:Y2=10:GOSUB50070
1110 X1=310:Y1=185:GOSUB50070:FORI=1TO500:NEXTI:GOSUB50030
1120 PRINT:PRINTTAB(5)"ELLIPSEN ZEICHNEN":FORI=1TO500:NEXTI:GOSUB50000
1130 XM=170:YM=150
1140 FORI=0TO16:WU=I*90:WO=WU+90
1150 HX=20+8*INT((3+I)/4):HY=10+8*INT((2+I)/4)
1160 GOSUB50080:NEXTI:X1=0:Y1=0:X2=319:Y2=0:GOSUB50060
1170 X2=0:Y2=199:GOSUB50060:X1=319:Y1=199:GOSUB50060:X2=319:Y2=0:GOSUB50060
1180 X1=100:Y1=0:X2=100:Y2=80:GOSUB50060:X1=0:Y1=80:GOSUB50060
1190 FORX=0TO100:Y=40+25*SIN(X/15):GOSUB50040:NEXTX
1200 X1=219:Y1=0:X2=219:Y2=80:GOSUB50060:X1=319:Y1=80:GOSUB50060
1210 XM=249:YM=40:HX=20:HY=30:WU=0:WO=360:GOSUB50080
1220 X1=279:Y1=10:X2=279:Y2=70:GOSUB50060:Y1=40:X2=309:Y2=10:GOSUB50060
1230 Y2=70:GOSUB50060
2000 RETURN
```

Listing 3. Grafik-Test und Demonstration

360 ist. Der Kreis ist ein Sonderfall der Ellipse. Dann muß nur HX = HY sein.

Für mathematisch Interessierte: Es werden die Parametergleichungen der Ellipse verwendet:
X = XM + HX*COS(WB) und
Y=YM + HX* SIN(WB)

Auch hier gibt es keine Einschränkung wie beim Strecken-Zeichnen in der Größe von XM, YM, HX, WU, WO.

W ist eine Laufvariable (ein Winkel) der in WB (gleicher Winkel im Bogenmaß) umgerechnet wird. L ist wieder die Löschmarke.

◆Ellipse löschen

Bis auf das Setzen der Löschmarke gilt dasselbe wie für das Zeichnen der Ellipse.

◆Kombination

Erfordert schon definierte Farbkennzahlen F1 und F2 (siehe Farbgebung) und schaltet dann die Hochauflösung an, löscht die Bit-Map und sorgt für die Farbe.

Soweit die Unterprogramme in Listing 2.

Ein Beispiel für die Möglichkeiten der Grafik-Bibliothek

Als ein Beispiel für die Möglichkeiten der Unterprogramm-Sammlung habe ich (ohne nun besonders auf Schönheit zu achten — das sind Sie ja von mir schon gewohnt), noch ein Hauptprogramm angefügt, mit dem Sie etwas herumprobieren können (Listing 3). Das Listing ist ausführlich kommentiert, so daß hier nur wenige Erläuterungen folgen müssen.

Beim Eintippen müssen Sie für einige Zeilen die Abkürzungen (siehe Handbuch Seite 130 ff) der Basic-Befehle eingeben, da die Zeilen sonst länger als 80 Zeichen werden.

Nach »RUN« sehen Sie ein Menü, das alle Möglichkeiten der Grafik-Unterprogramme ansteuert. Die Optionen 8 (Strecke zeichnen) bis B (Ellipse löschen) sowie 4 (Farbgebung) und 5 (Kombinationen) erfordern Eingaben. Es ist daher sinnvoll, diese Optionen nur im Normalmodus anzuwählen. Der Normalmodus ist immer dann zu erreichen, wenn Zeichenoperationen im Hochauflösungsmodus abgeschlossen sind.

Fortsetzung auf Seite 178

Computer bringen den K in Sch

Der VC 20 und der C 64 werden gerne als Spielzeug abgetan. Der Gegenbeweis liegt vor. Im Institut für Physiologische Chemie in München wurden fünf Commodore bei Untersuchungen im Labor eingesetzt.

Die Hormone aus den bunten »Töpfchen« werden in die Leber eingespritzt.

Gekachelte Wände, kalter Steinfußboden, Neonlicht, auf dem großen, grauen Arbeitstisch in der Mitte des Raumes ein Gewirr von Plastikschläuchen, an Stahlstangen aufgehängt. Große und kleine Plastikflaschen — etwas abseits Meßinstrumente und ein VC 20. Der erste Eindruck: steril und kompliziert — dennoch, der Blick durch die Glastür in dieses Labor der Abteilung Stoffwechselregulation des Instituts für Physiologische Chemie an der Uni München erweckt Neugier.

Franz M. Zwiebel, einer der wissenschaftlichen Mitarbeiter, öffnet einladend die Tür. Drei Schritte bis zu dem dominierenden Schlauchlabyrinth, und es wird noch rätselhafter: In den Schläuchen pulsiert eine klare Flüssigkeit. An dem Tisch sitzt ein junger Mann. Er könnte bestimmt erklären, was sich in dem Schlauchwirrwarr abspielt, wozu dieser Laboraufbau dient. Doch seine hochkonzentrierte Miene läßt keine Frage zu. Offensichtlich hat er überhaupt nicht bemerkt, daß jemand hereingekommen ist. Geduldig und präzise näht er an einem kleinen roten ovalen Etwas, das gut ausgeleuchtet im Scheinwerferlicht vor ihm an den Schläuchen baumelt. Es pulsiert rhythmisch. Erste Assoziation: ein rohes Stückchen Fleisch — nichts für jemanden mit schwachen Nerven. Der junge Mann, Thomas Kapsner, Medizinstudent im sechsten Semester, näht an einem pulsierenden Rattenherz.

Während der ganzen Zeit steht Franz M. Zwiebel schmunzelnd in einer Ecke des Labors. Er kennt die erstaunten und neugierigen Blicke der Outsider. Für ihn ist es Arbeitsalltag, was hier passiert. Wieder draußen auf dem Flur erklärt er: »Wir untersuchen Stoffwechselvorgänge an intakten Tier-Organen. Es ist der beste Weg, verläßliche Angaben über den menschlichen Stoffwechsel zu bekommen. Reagenzglasforschung bringt da nichts. Was Sie eben gesehen haben, ist ein isoliertes Rattenherz, das über einen künstlichen Kreislauf am Leben erhalten wird.« Aha, das leichte Schaudern vorhin war also berechtigt. Und die Schläuche mit der Flüssig-

reislauf
wung

keit stellen den künstlichen Kreislauf dar. Franz M. Zwiebel fährt fort: »Unsere Abteilung erforscht Stoffwechselvorgänge in der Zelle und den Transport von Stoffen über die Zellmembran. Das hört sich sehr kompliziert an, aber im Grunde genommen geht es darum, Wirkungen von Hormonen, anderen körpereigenen Stoffen sowie Medikamenten auf die Spur zu kommen.«

Der VC 20 regelt den Kreislauf

Das klingt alles recht einleuchtend, doch was soll der Commodore dabei? Franz Zwiebel schien die Frage erraten zu haben: »Der VC 20 ist für uns ein unentbehrlicher Helfer geworden. Er regelt den künstlichen Kreislauf bei unseren Experimenten. Nehmen wir als Beispiel das Herz. In einem lebenden Organismus pumpt ein Herz — je nach Belastung — unterschiedliche Blutmengen pro Zeiteinheit durch das Gefäßsystem. Dasselbe geschieht auch bei dem isolierten Herz: Mal kontrahiert der Muskel mehr, mal weniger. Folge ist, daß nicht zu jedem Zeitpunkt eine konstante Menge der Flüssigkeit aufgenommen und abgegeben wird. Dieser Flüssigkeit wollen wir eine bestimmte Konzentration des interessierenden Stoffes zusetzen. Um genaue Ergebnisse zu bekommen, muß die zugepumpte Menge der Durchflußgeschwindigkeit angepaßt werden. Schwankungen, die das lebende Organ bewirkt, sind über den Schrittmotor der Infusionspumpen zu regulieren. Und diese Regulation übernimmt der VC 20 für uns. Über eine Meßeinrichtung nimmt er die Menge der Flüssigkeit pro Zeiteinheit auf und bei Abweichungen vom Sollwert weist er den Schrittmotor an, schneller oder langsamer zu arbeiten.«

Ohne Computer geht nichts

Diese Erklärung vermittelt den Eindruck: Ohne Computersteuerung können die Stoffwechsel-Untersuchungen überhaupt nicht durchgeführt werden. Franz M. Zwiebel bestätigt: »Vorher wurden die Werte gemessen, Abweichungen sowie die notwendigen Neueinstellungen des Schrittmotors wurden mit Papier und Bleistift errechnet. Es war schon eine Revolution, als dafür ein Taschenrechner hergenommen werden konnte. Aber trotzdem, ehe man addiert, subtrahiert, dividiert, multipliziert und dann den Motor neu reguliert hatte, war oftmals schon zu viel Zeit vergangen. Eine Reihe von Untersuchungen brauchten erst gar nicht ausgewertet zu werden — die Arbeiten waren aufgrund der zu langen Verzögerungen wertlos.«

In einem anderen Labor läuft gerade ein Versuch mit einer Ratten-

Gewissenhaft näht Thomas Kaspner an einem intakten Rattenherz.

Leber. Auch hier wieder der bekannte Versuchsaufbau: Computer Schrittmotor, künstlicher Kreislauf. Mit höchster Aufmerksamkeit perfundiert die medizinisch-technische Assistentin Ursula Schwabe das Tier-Organ; das heißt in genau ausgeklügelten Zeitabständen spritzt sie Hormone ein.

Hier im Institut für Physiologische Chemie werden Grundlagen des Stoffwechsels erforscht. Sie sind Voraussetzung für weitere medizinische Forschungsprojekte. Es kann oft Jahre dauern, bis ein Patient von diesen Erkenntnissen profitiert.

Einer ist zu wenig

Ein Computer ist nur ein Tropfen auf den heißen Stein — nach diesem Motto schafften sich die Münchner gleich eine ganze Handvoll dieser Helfer für ihre Abteilung an. Alle gehören der Großfamilie Commodore an: zwei VC 20, ein C 64, ein 64 SX und ein cbm 8032. Auf den Geschmack ist man vor drei Jahren gekommen — kurz nach dem Erscheinen des VC 20 auf dem deutschen Markt wurde er gekauft. Die Ausstattung war damals sehr mager: 5

Computer bringen den Kreislauf in Schwung

KByte Arbeitsspeicher. Erst ein Jahr später gab es dann das dringend notwendige Zubehör: Speichererweiterung, Drucker, Laufwerke. Die Software schrieb Franz M. Zwiebel zum großen Teil selbst. Die Computer bewährten sich schnell. Ein Commodore kann nicht nur einen (künstlichen) Kreislauf zuverlässig regulieren, sondern er ist auch ein vortrefflicher »Rechenkünstler« und geduldiger Datenschlucker. Denn zur Auswertung der Daten, die bei den oftmals sehr langfristig angelegten Versuchen anfallen, wird er ebenfalls herangezogen.

Franz M. Zwiebel bei der Auswertung mit zwei seiner Helfer.

Franz M. Zwiebel schildert einen typischen Versuch: »Bei Untersuchungen von schnellen Transportvorgängen über die Zellmembran werden die Rohdaten wesentlich durch drei Faktoren mitbestimmt. Da ist die Größe des Gefäßraums und dessen Verzweigungen zu nennen sowie die Anzahl der Zellen. Wir brauchen aber unbedingt »saubere« Werte, die auch auf andere als die untersuchten Organe übertragbar sind. Es nützt nichts, zu wissen, wie schnell zum Beispiel Milchsäure im Stoffwechsel einer Versuchs-Leber verarbeitet wird, wenn man nicht gleichzeitig Angaben über den Gefäßraum und die übrigen beiden Faktoren machen kann. Deshalb geben wir bei jedem Versuch sogenannte Markierungstoffe in das zu untersuchende Organ, die entsprechende Rückschlüsse zulassen. Eine Substanz zur Markierung des Leber-Gefäßraums ist Zucker, im medizinischen Sprachgerauch Saccharose. Dieser Stoff geht nicht in die Zellen, nur in den Gefäß-Raum. Als Markierung für die Anzahl und Größe der Zellen nehmen wir Harnstoff. Er dringt in die ganze Leber ein. Die Werte für die interessierende Substanz, Milchsäure liegen zwischen den Werten für Saccharose und Harnstoff. Die Interpretation, wie schnell die Milchsäure in der Leber verarbeitet wird, können wir nur in Relation zu diesen Markierungsdaten vornehmen.«

PULS 96/TEIL C vom 7.12.83 ausgewertet am 28.2.84

ERGEBNISPROTOKOLL

BLATT 2

LFD.NR.	ZEIT (sec)	14C-SACCHAROSE <dPm/50 ul>	14C-HARNSTOFF <dPm/50 ul>	3H-LACTAT <dPm/50 ul>	TOH <dPm/50 ul>	3H-GLUCOSE <dPm/50 ul>
PULS C						
1	0.5	0	0	0	0	0
2	1	0	0	0	0	0
3	1.5	0	0	0	0	0
4	2	0.281	0	0	0	0
5	2.5	2.96	0	0	0	0
6	3	10.2	13.5	2.08	0	0
7	3.5	19.3	45.9	4.49	0	0
8	4	17.0	12.5	13.2	0	0
9	4.5	109	91.2	79.0	3.06	0
10	5	493	145	220	9.99	0

Teil einer Auswertung aus der Versuchsreihe »Puls«. Die Werte in Spalte 3 bis 7 sind Anzeichen dafür, welcher Anteil der Substanz zu der angegebenen Zeit (Spalte2) austritt.

Der Datenanfall pro Versuch ist enorm: bis zu 6000 Daten müssen in einigen Fällen verrechnet werden. Mit den Mikros lassen sich die notwendigen Umrechnungen der Rohdaten und die grafische Darstellung vornehmen. Werden umfangreichere Matrizen- oder Vektorrechnungen gefordert, ist der Commodore nicht mehr zuständig — dann läßt Franz M. Zwiebel die Daten auf dem Großrechner im Leibnitz-Rechenzentrum auswerten. Doch auch dann braucht er einen der Kleincomputer: Der Datentransfer geht über Lochstreifen oder Modem an das Rechenzentrum.

Auch bei der Ausbildung hilft der Computer

Regulation des künstlichen Kreislaufs während eines Versuchs, Auswertung der Daten nach dem Experiment — das sind längst nicht alle Einsatzgebiete der fünf Commodore-Computer. Franz M. Zwiebel verlangt noch ein bißchen mehr: So hat er Berichte- und sonstige Texte auf der Diskette abgespeichert. Die Schreibmaschine als »Textverarbeitungssystem« hat längst ausgedient.

Die Ausbildung von Studenten ist eine weitere Aufgabe, die zum Arbeitsalltag des eifrigen Münchner Forscherteams gehört. Und auch hier hat Franz M. Zwiebel ein sinnvolles Einsatzgebiet für seine Computer gefunden. Bei der Organisation des Unterrichts, der Vorbereitung und Auswertung von Prüfungsaufgaben hilft ihm ebenfalls die Commodore-Familie.

Wunschlos glücklich ist Franz M. Zwiebel aber noch nicht. Er »bastelt« an einer Vernetzung der Computer. Die Materie ist ihm inzwischen hinreichend vertraut — wer sich beruflich so viel mit Mikrocomputern beschäftigt will »der Sache auch auf den Grund gehen«. »Denn alles wird einfacher« prophezeit Franz M. Zwiebel, »wenn der VC 20 und der C 64 auf dieselben Datenbestände zugreifen können«. (kg)





WETTBEWERB:

```
50 REM *******************************
52 REM ***     SPRITE WETTBEWERB     ***
54 REM ***        RALPH MEYER        ***
56 REM ***    STOERTEBEKERWEG 32     ***
58 REM ***     2104 HAMBURG 92       ***
60 REM *******************************
70 REM
72 REM
75 :
80 REM *** DATA EINLESEN ***
90 :
100 FORI=0TO4
110 FORJ=0TO62
120 :READBY:POKE250*64+64*I+J,BY
130 NEXTJ,I
210 FORI=1TO8:READSN(I):NEXTI
300 :
1000 REM *** SPRITE PARAMETER SETZEN **
1005 :
1010 VIC=53248
1020 POKEVIC+32,11:POKEVIC+33,11
1030 POKEVIC+21,1:REM SPRITE 0 AN
1040 POKEVIC+28,1:REM MULTICOLOR SPR
1050 POKEVIC+39,0:REM SC=SCHWARZ
1060 POKEVIC+37,8:REM MC0=ORANGE
1070 POKEVIC+38,2:REM MC1=ROT
1075 POKEVIC+1,100:NR=1
1500 :
2000 REM *** SPRITE BEWEGEN VON
2002 REM *** LINKS NACH RECHTS
2008 :
2010 PRINT"♥"
2020 FORX=1TO255STEP2
2030 POKEVIC+0,X
2040 POKE2040,250+SN(NR)
2050 NR=NR+1
2060 IFNR>8THENNR=1
2070 FORI=1TO50:NEXT
2080 NEXTX
2999 END
4000 :
5000 REM *** DATEN FUER DIE 5 BEWEGUNGS
5002 REM *** PHASEN DER FIGUR
5004 :
9900 REM *** SPRITE 0
9940 DATA 2, 34, 0, 10, 168, 0, 34, 148, 0
9950 DATA 2, 89, 64, 10, 85, 80, 8, 69, 64
9960 DATA 34, 64, 0, 0, 85, 0, 3, 245, 64
```

Viele Leser werden schon auf die Gewinner unseres Wettbewerbs gewartet haben.

Hier sind sie, die schönsten Sprites.

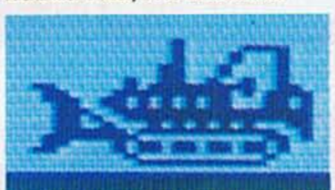


Peter Nick, 6540 Simmern

(Pumuckl)

Sie können sich vorstellen, daß uns eine Entscheidung nicht leichtgefallen ist. Denn es haben sich viele Leser an diesem Wettbewerb beteiligt und ein Sprite sah schöner aus als das andere. Schließlich mußte sich jeder Redakteur vor den Monitor

Alexander Glaser, 6113 Babenhausen



(Raupe)

setzen und zu jedem Sprite seine eigene Beurteilung abgeben. Zum Schluß wurden die Beurteilungen ausgewertet, dann stand der Sieger fest. Das Multicolor Sprite von Ralph Meyer aus Hamburg bekam die meisten Stimmen. Es ist eines der wenigen Sprites, die in Multicolor erstellt wurden und sieht doch recht hübsch aus.

Torsten Ludwig, 2857 Sievern



(Fuchs)

Wenn Sie das beistehende Listing abtippen und das Programm starten, bemerken Sie auch die Liebe fürs Detail, die in jeder Bewegung des kleinen Männchens steckt.

Ähnliches gilt auch für das Motorrad, das Erwin Schaal aus Auenwald schickte. Nur ist es kein Multicolor-Sprite



Joachim Goebel, 6203 Hochheim

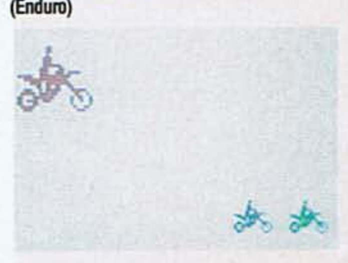
(Cowboy)

und kam deshalb auf den zweiten Platz. Multicolor Sprites haben zwar den Nachteil, daß die Auflösung des Sprites geringer ist, es wird gröber, aber es ist etwas schwieriger zu programmieren.

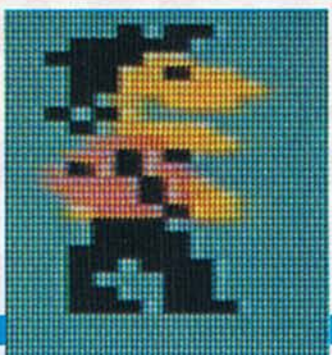


3. Preis: Stefan Hübner, 3000 Hannover

Thomas Reuter, 7742 St. Georgen/Schw. (Enduro)









1. Preis: Ralph Meyer, 2104 Hamburg

```
9970 DATA 43, 184, 0, 191, 191, 208, 255, 239
, 212
9980 DATA 95, 250, 148, 67, 252, 0, 0, 168, 0
9990 DATA 34, 170, 0, 42, 186, 128, 42, 226,
128
10005 DATA 170, 2, 128, 160, 2, 160, 128, 0,
168
11000 REM *** SPRITE 1
11010 DATA 34, 40, 128, 8, 170, 0, 10, 148, 0
11020 DATA 170, 153, 64, 10, 85, 80, 8, 85, 6
4
11030 DATA 42, 64, 0, 128, 85, 0, 3, 245, 64
11040 DATA 3, 252, 0, 3, 236, 0, 31, 190, 208
11050 DATA 94, 255, 208, 67, 252, 0, 0, 168,
0
11060 DATA 0, 168, 0, 10, 170, 0, 170, 10, 12
8
11070 DATA 168, 2, 136, 160, 2, 160, 160, 0,
160
12000 REM *** SPRITE 2
12010 DATA 34, 34, 0, 10, 168, 0, 10, 148, 0
12020 DATA 42, 89, 64, 10, 85, 80, 40, 69, 64
12030 DATA 10, 64, 0, 32, 85, 0, 3, 245, 64
12040 DATA 3, 232, 0, 3, 236, 0, 7, 188, 0
12050 DATA 23, 189, 0, 22, 253, 0, 0, 168, 0
12060 DATA 2, 168, 0, 10, 168, 0, 10, 42, 0
12070 DATA 8, 10, 0, 10, 10, 0, 10, 138, 128
13000 REM *** SPRITE 3
13010 DATA 0, 138, 0, 10, 168, 0, 42, 148, 0
13020 DATA 10, 89, 64, 10, 85, 80, 8, 69, 64
13030 DATA 34, 64, 0, 8, 85, 0, 3, 245, 64
13040 DATA 15, 184, 0, 59, 188, 0, 63, 237, 0
13050 DATA 15, 237, 64, 7, 249, 64, 2, 160, 0
13060 DATA 2, 168, 0, 10, 168, 0, 10, 42, 0
13070 DATA 8, 10, 0, 10, 10, 0, 10, 138, 128
14000 REM *** SPRITE 4
14010 DATA 2, 34, 0, 10, 168, 0, 138, 148, 0
14020 DATA 42, 153, 64, 10, 85, 80, 40, 85, 6
4
14030 DATA 138, 64, 0, 32, 69, 0, 3, 245, 64
14040 DATA 3, 188, 0, 3, 188, 0, 3, 188, 0
14050 DATA 3, 156, 0, 3, 212, 0, 0, 148, 0
14060 DATA 0, 168, 0, 0, 168, 0, 0, 40, 0
14070 DATA 0, 40, 0, 0, 168, 0, 0, 170, 0
20000 REM *** REIHENFOLGE ***
20100 DATA 0,3,4,2,1,2,4,3

READY.
```

Auf eine ganz andere Idee kam Stefan Hübner aus Hannover. Die an sich langweilige Computerschrift läßt sich mit seinem Sprite sehr gut verzieren. Etwa als Anfangsbuchstabe eines Absatzes.



(»G«)

Das verzierte »G« von Gandalf aus dem »Hobbit« brachte ihm den dritten Platz ein.

Salomonisches Urteil

Als wir begannen, die Lesereinsendungen zu prüfen und zu bewerten, merkten wir, daß es fast unmöglich sein würde, einen Sieger zu erhalten, dessen Sprite als einziges den Gesamtpreis von 1000 Mark rechtfertigen würde. Zu gering waren die Unterschiede. Deshalb entschieden wir uns für eine gerechtere Verteilung des Gesamtpreises. Der Sieger erhält 350 Mark, der zweite Platz, das Motorrad, fährt mit 200 Mark nach Hause und das hübsche »G« wird mit 100 Mark honoriert. Da die restlichen Sprites unserer TOP-TEN auch sehr hübsch sind und auch nicht wesentlich schlechter ausfallen, beschlossen wir, jedem von ihnen eine Prämie von je 50 Mark zukommen zu lassen.



Pavel Zubec, 4900 Herford

(Supermann)

Martin Kronbuegel, Hamburg 92



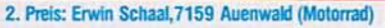
(Segelschiff)

nicht aus einem, sondern aus mehreren Sprites. Deshalb kamen sie auch nicht unter die ersten drei Gewinner. Wir hoffen, daß die Leser und auch die Gewinner mit dieser »salomonischen« Regelung einverstanden sind.

(Die 64'er Redaktion)

Dem aufmerksamen Beobachter wird auffallen, daß zwei der abgebildeten Sprites sich von allen anderen unterscheiden. Und zwar sind das der Pumuckl und der Supermann. Diese Sprites sehen zwar sehr schön aus, bestehen aber leider





2. Preis: Erwin Schaal, 7159 Auenwald (Motorrad)



Nachdem wir ein Magazin speziell für die Commodore-Computer geschaffen haben, rückt ein Wunschtraum in greifbare Nähe:
Wir wollen eine Programmbibliothek aufbauen. Eine Bibliothek, zu der jeder Leser einen (oder auch beliebig viele) Beiträge leisten kann.

Warum eine Bibliothek?

In fast jeder Computerzeitschrift werden irgendwann einmal kleine Programme, Utilities, abgedruckt, die jeder Programmierer sehr gut brauchen kann. Aber wer kann sich schon jede Zeitschrift jeden Monat leisten. Und nicht jeder hat die Zeit, jedes eigentlich interessante Programm abzutippen. Deshalb wollen wir eine Sammlung von Unterprogrammen zusammenstellen, die jedem Leser zur Verfügung steht.

Was soll die Bibliothek enthalten?

Sie soll hauptsächlich Programme enthalten die man entweder in bestehende Programme einsetzt oder von Anfang an in ein zu entwickelndes Programm einbaut.
Dazu gehören zum Beispiel: Druckerhilfsprogramme, wie Hardcopy-Routinen, Unterprogramme für Plotter, Grafik-Unterprogramme (wie zum Beispiel der Draw-line Algorithmus aus dem 64'er/4), Unterprogramme für Bildschirmaufbau, Formatierungshilfen (Print Using oder Print At, Maskenaufbauprogramme, etc.), Sortierroutinen. Programme zur Anpassung bestimmter Peripherie an den Computer (zum Beispiel Epson an C 64)

Auch Utilities, wie etwa Renumber, Merge, Delete oder Copy finden ihren Platz. Ansonsten sind Ihrer Fantasie keine Grenzen gesetzt. Machen Sie Vorschläge. Es sollen die Leser, also Sie sein, die die Bibliothek aufbauen. Um diese Bibliothek aufbauen zu können, müssen jedoch einige Regeln beachtet werden.

Ein Problem: die Variablen

Diese Unterprogramme sollen von jedem eingesetzt werden können, und das ohne allzugroßen Änderungsaufwand. Ein Problem dabei sind die verwendeten Variablen. Wenn man ein Bibliotheksprogramm in sein eigenes Programm integrieren will, muß man sich darauf verlassen können, das keine gleichen Variablen für unterschiedliche Zwecke benutzt werden.

Deshalb bestimmen wir die Namen der Variablen, die Sie für Ihre Unterprogramme, die in unsere Bibliothek kommen sollen, benutzen dürfen. Die Art der Parameterübergabe erklären wir dann später noch.

Am besten benutzen wir in unseren Unterprogrammen sonst selten benutzte Variablennamen:
Q1 bis Q9 oder Q9(0) bis Q9(10) für numerische Variable
Y1$ bis Y9$ oder Y9$(0) bis Y9$(10) für String Variable
W1(i,j) bis W9(i,j) für numerische Felder
V1$(i,j) bis V9$(i,j) für String Felder

Als Laufvariable sollten Sie die sonst üblichen nehmen, aber in jeweils doppelter Ausführung:
II,JJ,KK,LL,MM,NN

Beispiel für Parameterübergabe

Angenommen, es existiert folgendes Unterprogramm (siehe Listing). Dann kann dieses Unterprogramm durch das Hauptprogramm folgendermaßen aufgerufen werden:

```
7120 ......
7130 ....
7140 Q1=K
7150 GOSUB 100:REM SORTIEREN FF$
7160 ...
```

An diesem Beispiel kann man gleich zwei Punkte erkennen:

1. In Zeile 7140 wird die Variable K an die Variable Q1 übergeben. K und Q1 haben jetzt also den gleichen Wert, nämlich die Größe des Feldes FF$(). Nur derjenige, der das Unterprogramm in sein Programm einbauen will, muß wissen, welche Variable er übergeben muß. Denjenigen, der das Unterprogramm geschrieben hat, interessiert das überhaupt nicht. Der muß dem späteren Benutzer lediglich mitteilen, welche Variablen er benutzt und was sie bedeuten.

2. Das Feld FF$( ). Der Ersteller des Unterprogramms hat nicht den Namen FF$ gewählt, sondern V1$ (gemäß unseren Konventionen siehe oben). Erst der spätere Benutzer hat den Namen (gemäß seinen Bedürfnissen) in FF$ umgewandelt. Und das aus folgendem Grund:

Eine Parameterübergabe eines Feldes ist nicht nur zeitraubend, sondern auch der Speicherplatz des Feldes muß verdoppelt werden (also auch neu dimensioniert werden). Das ist hier die bedeutendste Einschränkung in Basic. Aus diesem Grunde sollte in diesem Fall die Möglichkeit bestehen, das Unterprogramm zu ändern (also bitte keinen Listschutz oder ähnliches einbauen). Die Zeilennummern sollten in Zehnerschritten durchnummeriert werden.

Aufbau des Unterprogramms

Bitte schicken Sie Ihre Programme mit diesem Aufbau ein: In der obersten Zeile die Bezeichnung UP gefolgt vom Namen (im Beispiel Zeile 110). Darunter die Übergabeparameter (130-140). Durch einen Strich getrennt folgen die im Unterprogramm verwendeten Variablen ebenfalls mit einer kurzen Erklärung (170-180). Am Schluß des Programmkopfes soll eine kurze Beschreibung des Programms stehen (200-220). Erst danach folgt das eigentliche Programm. Es sollte mit RETURN beendet werden.

Daß zusätzlich noch eine ausführliche Programmbeschreibung (Sinn und Zweck, Funktionsweise, Variablentabelle etc.) vorhanden sein muß, ist eigentlich selbstverständlich. Gerade bei Utilities ist eine ausführliche Bedienungsanleitung (auch für Anfänger verständlich) unbedingt notwendig! Eine gute Lösung ist es, diese Beschreibungen zusätzlich in REM-Zeilen an das Programm anzuhängen (hinter dem RETURN). Man kann sie dann bei Bedarf einfach listen und auch, wenn nicht mehr benötigt, einfach löschen.

Wir werden alle eingeschickten Programme testen und jeden Monat die besten prämieren, veröffentlichen und zusätzlich als Teil eines Programmpaketes als Leser-Service auf Diskette anbieten. Wir freuen uns über jeden Beitrag. Jeder kann gewinnen, es werden jeden Monat mehrere Preise vergeben. Und jeden Monat wird ihr Unterprogramm wieder mitberücksichtigt. Schicken Sie Ihre Unterprogramme an: Verlag Markt & Technik, Redaktion 64'er, Programmierwettbewerb: Programmbibliothek, Hans-Pinsel-Straße 2, 8013 Haar bei München.

Fortsetzung von Seite 169

Drücken Sie dann "2", sind Sie wieder im normalen Rechnerbetrieb.

Sollten Sie durch irgendeinen Umstand (zum Beispiel durch Drücken der »RUN/STOP«-Taste) im Hochauflösungsmodus aus dem Programm fallen, dann hilft der folgende Weg:

1. »SHIFT« + »CLEAR/HOME«
2. »RUN« »RETURN«
3. dann »2« eingeben

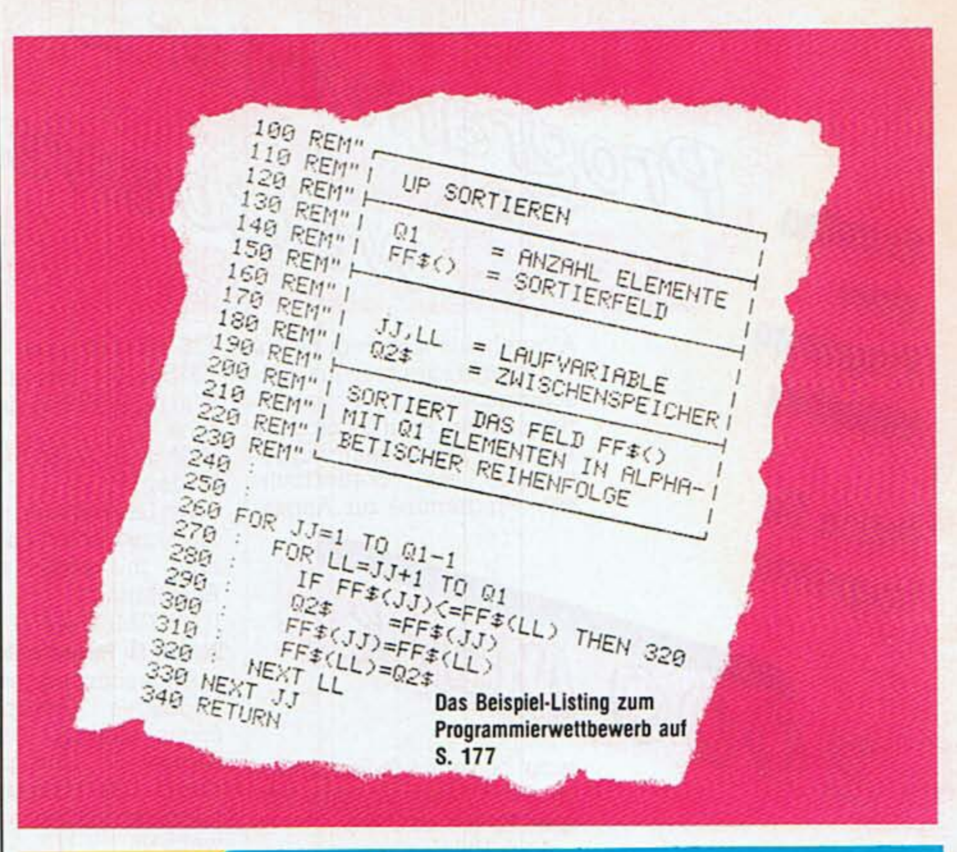
Die Option »C« zeigt eine kleine Demonstration von Möglichkeiten der Grafik-Unterprogramme. Allerdings sollten Sie ein bißchen Zeit mitbringen, wenn Sie C anwählen: Das ganze dauert zirka 25 Minuten.

Option 6 (Punkte zeichnen) ist so eingerichtet, daß 320 Punkte in Form einer Sinus-Funktion gezeichnet und mit Option 7 (Punkte löschen) teilweise wieder gelöscht werden.

Ausblicke — schnellere Grafik durch Maschinensprache

Diese Folge soll nicht beendet werden, ohne einen kleinen tröstlichen Ausblick. Wie Sie — besonders im letzten Programm — feststellen konnten, braucht man schon einiges Sitzfleisch für hochauflösende Grafik in Basic. Wenn Sie aber ein kommerzielles Grafik-Programm laufen sehen, geht das alles erheblich schneller. Was ist der Unterschied? Da wäre zunächst einmal die Programmiersprache: Unser C 64 kann eigentlich gar kein Basic. Er braucht den Basic-Interpreter, der zunächst jeden Befehl liest und dann in Maschinensprache übersetzt. Die versteht unser Rechner zwar, die Übersetzung und das Lesen dauern jedoch lange Zeit. Eine starke Beschleunigung der Grafik ist möglich durch Programmieren in Maschinensprache. Einige solche Maschinenspracheprogramme zur beschleunigten Grafik werden in den nächsten Folgen gezeigt. Allerdings stoßen wir da bald an die Grenzen unseres Commodore. Ein 8-Bit-Computer mit zirka 1 Megahertz Taktfrequenz wie unser C 64 ist beispielsweise in der Fließkomma-Arithmetik (wie sie für das Zeichnen von Ellipsen nötig ist) zeitlich gehandicapt, und deswegen sind der Geschwindigkeit bei komplexerer Grafik doch einige Grenzen gesetzt.

(Heino Ponnath)



```
100 REM"┌─────────────────────────────┐
110 REM"| UP SORTIEREN                |
120 REM"├─────────────────────────────┤
130 REM"| Q1                          |
140 REM"| FF$()   = ANZAHL ELEMENTE   |
150 REM"├         = SORTIERFELD       ┤
160 REM"|                             |
170 REM"| JJ,LL   = LAUFVARIABLE      |
180 REM"| Q2$     = ZWISCHENSPEICHER  ┤
190 REM"├─────────────────────────────┤
200 REM"| SORTIERT DAS FELD FF$()     |
210 REM"| MIT Q1 ELEMENTEN IN ALPHA-  |
220 REM"| BETISCHER REIHENFOLGE       |
230 REM"└─────────────────────────────┘
240 :
250 :
260 FOR JJ=1 TO Q1-1
270 :  FOR LL=JJ+1 TO Q1
280 :   IF FF$(JJ)<=FF$(LL) THEN 320
290 :   Q2$    =FF$(JJ)
300 :   FF$(JJ)=FF$(LL)
310 :   FF$(LL)=Q2$
320 :  NEXT LL
330 NEXT JJ
340 RETURN
```

Das Beispiel-Listing zum Programmierwettbewerb auf S. 177